ABC INFORMATIQUE

N° 81

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

L'informatique à l'école

Les pinces du robot

Mémoire : Wafadrive de Rotronics

Bloc mémoire du SE du C64

EDITIONS ATLAS

M 6062-81-12,50 F 90 FB - 3,80 FS - $ 1.95

Page manquante
(publicité et colophon)

Langues modernes

La récente introduction de l'informatique à l'école suscite à la fois inquiétude et enthousiasme. Les questions sont posées.

Q Pourquoi l'enseignement de la programmation informatique à des élèves du primaire suscite-t-elle tant de discussions ?

R L'enseignement traditionnel faisait jouer un rôle entièrement passif à l'enfant. Lui donner un rôle actif engendre immanquablement des inquiétudes ; il est donc peu surprenant que l'introduction de la programmation en milieu scolaire soit considérée comme une aventure présentant des risques autant pour les professeurs que les élèves.

L'informatique fait partie des programmes scolaires depuis déjà quelques années et séduit de plus en plus de parents et d'élèves. Des études menées auprès de ceux-là au début des années quatre-vingts ont démontré qu'ils considèrent l'informatique comme la troisième matière la plus importante, après les mathématiques et le français ; mais cet enthousiasme n'est pas toujours partagé par les enseignants ni par les recruteurs de l'industrie informatique.

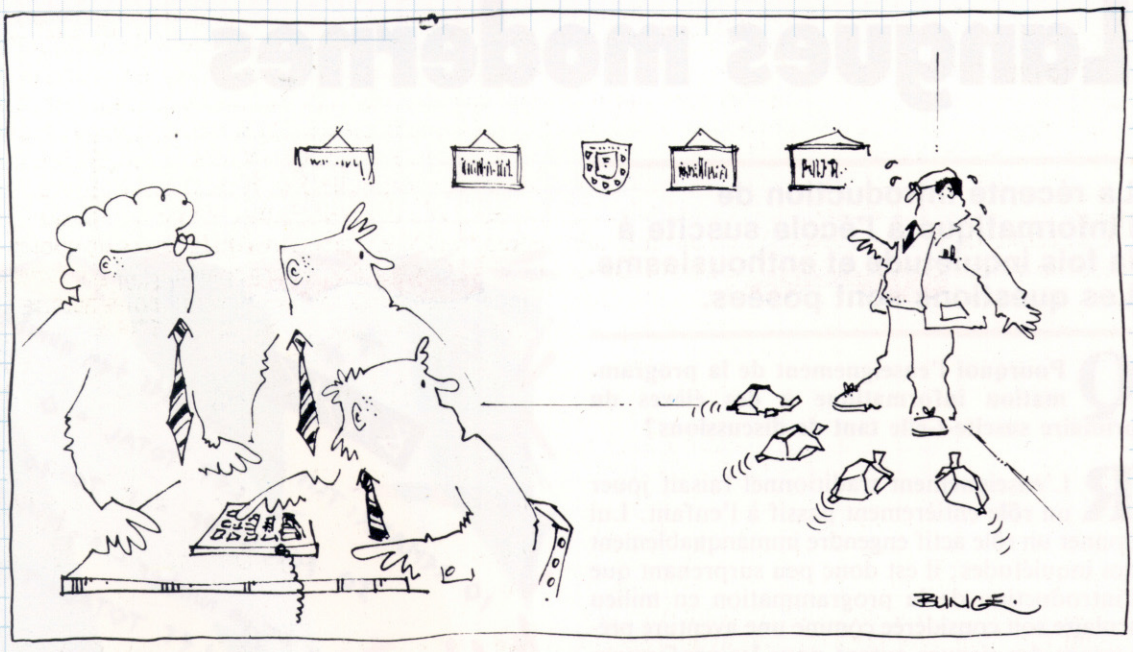
Certains enseignants croient, par exemple, qu'il serait préférable que l'informatique s'insère dans l'ensemble des matières plutôt qu'elle ne constitue une matière distincte. Cela entraîne des difficultés pour la discipline informatique, surtout au niveau des problèmes financiers que doivent affronter la plupart des écoles.

Qu'un enfant ne se destine pas à être artiste ou écrivain professionnel ne justifie pas qu'il soit privé de l'enseignement des arts et des lettres. Comme nous l'avons déjà souligné dans cette suite d'articles, la programmation aide les jeunes enfants à acquérir des notions considérées normalement comme hors de leur portée ; cette constatation à elle seule justifie l'enseignement de la programmation à l'école.

Cet enseignement a été, jusqu'à présent, influencé par le fait que la plupart des machines sont livrées avec le BASIC. En raison de sa nature abstraite, le BASIC ne peut être assimilé par des enfants de moins de 12 ans. Aussi, dans certains

Autres langages

La prise en charge de divers langages sur les micros contribue à répandre l'enseignement de la programmation dans les écoles. Certains langages, comme PROLOG, peuvent être généralement étudiés du seul point de vue de la structure qu'ils imposent. D'autres, comme COMAL et LOGO, offrent une introduction plus disciplinée à la programmation que BASIC (bien qu'il soit facile à apprendre, il encourage facilement les abus et peut conduire à l'adoption de mauvaises techniques de programmation. (Cl. Alan Adler.)



cas, les cours d'informatique ont-ils été dispensés uniquement dans des écoles secondaires. De plus, bien que très pratique à de nombreux égards, le BASIC ne présente que peu d'intérêt dans la manière d'enseigner en raison de son approche non structurée et en raison des mauvaises habitudes de programmation qu'il entraîne pour les débutants. Malheureusement, de nombreux professeurs ont abordé eux-mêmes le sujet en donnant des cours de BASIC, propageant ainsi sa mauvaise influence malgré la disponibilité d'autres langages comme LOGO, COMAL, PROLOG.

Q Si, comme on l'admet généralement, l'utilisation du BASIC donne de mauvaises habitudes de programmation, existe-t-il d'autres langages ?

R L'introduction de versions BASIC plus structurées a contribué à résoudre certaines lacunes des matériels, mais la programmation éducative se situe principalement au niveau des trois langages que nous venons de mentionner. Nous avons déjà parlé du LOGO de façon détaillée. On encourage actuellement son utilisation dans l'enseignement aux États-Unis, en Grande-Bretagne comme en France.

Dans un établissement d'État danois, Borge Christensen a mis au point COMAL (COMmon Algorithmic Language). COMAL, selon son créateur, est une forme structurée du BASIC. Christensen et ses collègues estimaient que les programmes BASIC de leurs étudiants étaient si mal écrits qu'ils étaient presque incompréhensibles. « Après avoir exécuté des milliers de programmes BASIC, j'ai finalement compris que ce langage était vraiment inadéquat » dit-il. Les arguments de Christensen en faveur de COMAL réussirent à influencer le gouvernement danois à un point tel que ce dernier décida de faire de COMAL le premier langage de programmation dans les écoles. On l'a préféré au BASIC en Irlande; il suscite déjà beaucoup d'intérêt ailleurs.

PROLOG (PROgramming in LOGic) a été utilisé par de petits groupes d'enseignants. Au moyen de PROLOG, un problème exprimé en termes logiques peut être facilement traduit en un programme informatique. Dans certaines écoles, on a mis sur pied un projet visant à mettre au point des matériels destinés à enseigner la logique comme discipline de programmation à des élèves de 10 à 13 ans. Par la suite, certains de ces élèves devinrent de très bons programmeurs, et d'autres réussirent les tâches données en classe mais ne firent que très peu de travail par eux-mêmes. Le projet a remporté plus de succès auprès des dernières classes. Normal, puisque l'informatique est de plus en plus répandue à l'école. Le projet visait surtout l'enseignement de la logique et non de la programmation.

Q Qu'est-ce qui permet à ces langages d'enseigner facilement la programmation à un enfant ?

R LOGO a fait l'objet de recherches intensives pour répondre aux personnes qui déclaraient ne pas croire aux avantages que la programmation pouvait apporter aux enfants. (Malheureusement, le faible nombre d'écoles utilisant COMAL et PROLOG n'a pas justifié d'étude appropriée de ces langages.) Le US National Science Foundation a lancé le projet Brookline LOGO pour étudier l'utilisation de LOGO par des enfants. La première phase du projet s'adressait à des élèves âgés de 11 ans dans un laboratoire informatique et avait pour but d'observer leurs diverses activités. La seconde phase introduisait des ordinateurs en milieu scolaire et avait en partie pour but de mettre au point des matériels adaptés aux diverses matières. Un résultat intéressant fut que tous les enfants réussirent très bien leurs travaux LOGO, quelle qu'ait été leur apti-

tude dans les autres sujets. Ce résultat s'est aussi répété dans d'autres projets LOGO.

Q A-t-on réalisé d'autres études pour découvrir les effets de l'apprentissage de la programmation sur les enfants ?

R Le New York Academy of Science et trois écoles de la ville de New York ont mis sur pied un projet d'enseignement de LOGO. On a installé des ordinateurs dans les classes ; des visites hebdomadaires étaient faites par les organisateurs du projet, et des enseignants volontaires participaient à des séminaires mensuels. Le personnel a estimé que le projet était une réussite dans 95 % des classes. Le directeur du projet, Michael Tempel, a souligné l'existence de nombreuses ressemblances avec le projet Brookline. La présence de LOGO dans les classes eut pour principale conséquence de susciter une interaction positive parmi les enfants. Ceux-ci présentaient des difficultés dans d'autres matières, réussirent très bien avec LOGO. « Les avantages sur le plan pédagogique sont évidents, dit-il, les enfants sont ainsi engagés dans de réels processus intellectuels et sociaux. » A la suite de ce projet, on décida d'installer une école pilote dans la région, qui servira de centre de recherche LOGO.

Un projet s'étendant sur trente mois, mené au Center for Children and Technology à New York, a tenté d'évaluer la possibilité d'appliquer à d'autres matières ces stratégies de résolution de problème, et d'évaluer l'effet des travaux LOGO sur l'interaction sociale. Le rôle important de la planification dans les processus d'apprentissage est de plus en plus reconnu, et des études ont démontré que les enfants présentent souvent des lacunes à ce niveau. La nature structurée de LOGO encourage le fractionnement des problèmes de programmation en plus petites unités pendant l'étape de planification. Les recherches se concentrent surtout sur les effets qu'a la programmation sur l'acquisition de ces méthodes essentielles d'apprentissage.

Il est intéressant d'observer comment les ordinateurs ont encouragé la coopération parmi les étudiants. « Il y a plus d'interactions reliées au travail pendant les activités informatiques que pendant toute autre activité menée de façon autonome en classe », a remarqué le professeur. « De plus, il y avait plus de discussions associées au travail que pendant les autres activités. » Une coopération et une collaboration similaires se sont aussi manifestées à l'école Lamplighter au Texas, à laquelle Texas Instrument a offert cinquante micros dotés du LOGO TI.

Q Quels problèmes ont gêné l'efficacité de ces études ?

R Les problèmes rencontrés au cours des recherches relatives à l'utilité de la programmation en milieu scolaire sont très nombreux. Les responsables de l'enseignement hésitent à réserver beaucoup de temps à la programmation, alors qu'on n'a pas encore réussi à prouver qu'il pouvait être avantageux de l'inclure dans les matières obligatoires. Il y a des problèmes de financement et de disponibilité du matériel. Il est également difficile d'organiser une étude contrôlée, puisque la création d'un groupe de contrôle prive d'autres enfants d'heures d'utilisation de l'ordinateur, tandis que la création d'un groupe de contrôle dans une autre école complique la comparaison en raison des différences de contextes et de méthodes d'enseignement. De nombreux professeurs ne connaissent pas mieux l'informatique que les enfants ; cette situation explique que souvent les enfants ne peuvent pas tirer parti d'une véritable source de connaissance.

Q Outre le développement de nouvelles approches logiques au travail, quelles autres applications pratiques peuvent résulter de l'apprentissage de la programmation chez un enfant ?

R Dès qu'un enfant a pris contact avec l'art de la programmation, le nombre d'applications potentielles est énorme. Un groupe d'enfants a réussi à utiliser des tortues pour réaliser une chorégraphie : en programmant le mouvement des tortues, ils ont traduit leurs déplacements en danse. Les caractéristiques de LOGO ont été utilisées pour susciter des histoires dans des cours de langue. On propose aux élèves un carrefour illustrant divers choix qui déterminent le cours que prendra l'histoire. De plus, LOGO est doté de fonctions sonores très complètes. Il est ainsi facile de définir des notes, des accords et des registres, ce qui permet aux enfants d'utiliser l'ordinateur pendant leurs cours de musique.

Q Si peu d'écoles enseignent la programmation, et si elles le font dans des langages différents, comment assurer la diffusion de ce phénomène ?

R Le choix de LOGO comme langage de programmation pour de jeunes enfants a ouvert de nouvelles possibilités en matière d'enseignement. Mais le BASIC règne toujours au niveau du secondaire, et l'on doit veiller à ce qu'un support continu pour LOGO soit étendu à tout le système éducatif. Sinon, le travail accompli dans les classes du primaire avec LOGO sera compromis par le monopole du BASIC dans les niveaux suivants. Comme c'est le cas pour les ordinateurs eux-mêmes, l'utilisation de l'ordinateur doit être standardisée pour qu'elle soit efficace et bénéfique.

L'apprentissage de la programmation donne un outil puissant aux élèves, met à leur disposition un autre moyen d'expression et de création et leur permet d'acquérir une pensée structurée et précise. Enfin, il facilite la compréhension dans d'autres matières. Bien que d'autres recherches s'imposent dans ce domaine, les résultats obtenus grâce à l'apprentissage de la programmation sont positifs et très encourageants.

La clé des champs

Les enregistrements contiennent des zones de données de types divers. Pour en faire usage, voyons ce qu'on entend par « clé primaire » et « clé secondaire ».

Une « clé » est un outil dont dispose le logiciel de gestion de données afin de localiser un enregistrement spécifique. Prenons l'exemple du manuel de maintenance d'une automobile. Vous voulez procéder au réglage du carburateur : la façon de faire peut être précisée quelque part dans l'ouvrage, mais où ? Supposons que ce soit page 36 : le numéro de page sera donc la clé de l'information que vous recherchez. Mais pour cela, il faut déjà avoir repéré le passage en question... Vous préférerez sans doute consulter l'index, qui vous renverra page 36, sans que vous soyez contraint de feuilleter tout le manuel.

La gestion d'une base de données obéit aux mêmes principes. Chaque enregistrement d'un fichier donné reçoit un numéro qui lui est propre, et qu'on appelle *clé primaire*. Cependant, pour accéder à un enregistrement particulier, vous devrez les consulter tous séquentiellement jusqu'à ce que vous tombiez sur le bon, ou bien, si vous connaissez déjà son numéro, vous pourrez aussitôt l'extraire du fichier. Il est également possible de faire usage de l'une des zones comme clé (ce qu'on appelle une *clé secondaire*). Revenons à notre exemple de manuel automobile ; la zone NOMDEPIÈCE va nous servir de clé.

La plupart des logiciels de gestion de données autorisent l'emploi de zones spécifiques comme ZONES-CLÉS. Lorsque l'une d'elles est ainsi sélectionnée (NOMDEPIÈCE dans notre exemple), le programme de gestion garde en mémoire une table des mots (chaînes de caractères) contenus dans cette zone, ainsi que le numéro d'enregistrement (clé primaire) correspondant. Lorsque vous partez à la recherche de l'enregistrement concernant les carburateurs, le logiciel analyse la table NOMDEPIÈCE$ jusqu'à ce qu'il trouve la chaîne « carburateur » ; il voit ensuite quel est le numéro d'enregistrement qui lui correspond, et extrait de l'ensemble le renseignement demandé. Voici comment vous pourriez procéder si vous aviez à écrire en BASIC un gestionnaire de données très simple :

```
INPUT "ENTREZ LA ZONE-CLÉ"; CLÉ$
INPUT "ENTREZ LE MOT À RECHERCHER"; MOT$
GOSUB 20000 : REM SOUS-PROGRAMME RECHERCHE
PRINT RÉSULTREC
```

Que se passe-t-il ici ? Un des nombreux tableaux a été sélectionné grâce à CLÉ$, puis parcouru par le sous-programme, la chaîne de caractères MOT$ servant de clé de recherche. Si ce sous-programme est suffisamment puissant et détaillé, il sera même capable de prendre en compte certaines erreurs de frappe et de trouver quand même l'enregistrement qu'il faut. Notre routine reste très simple et n'aura besoin que de parcourir tous les enregistrements, sans avoir à garder en mémoire une table de toutes les clés.

Les premiers logiciels de gestion de bases de données tournant sur micro-ordinateurs restaient très sommaires ; il était possible d'en écrire un soi-même en BASIC. L'apparition du QL de Sinclair, pourvu d'un microprocesseur 68000, a changé la situation. La firme Psion, faisant usage de mini-ordinateurs Vax, mit au point le logiciel Archive, offrant ainsi à l'usager moyen des capacités très sophistiquées. Pour voir comment il fonctionne, créons une base de données extrêmement simplifiée ne comportant que quatre enregistrements, dont chacun ne sera doté que de deux zones, NOMDEPIÈCE et MISEENŒUVRE :

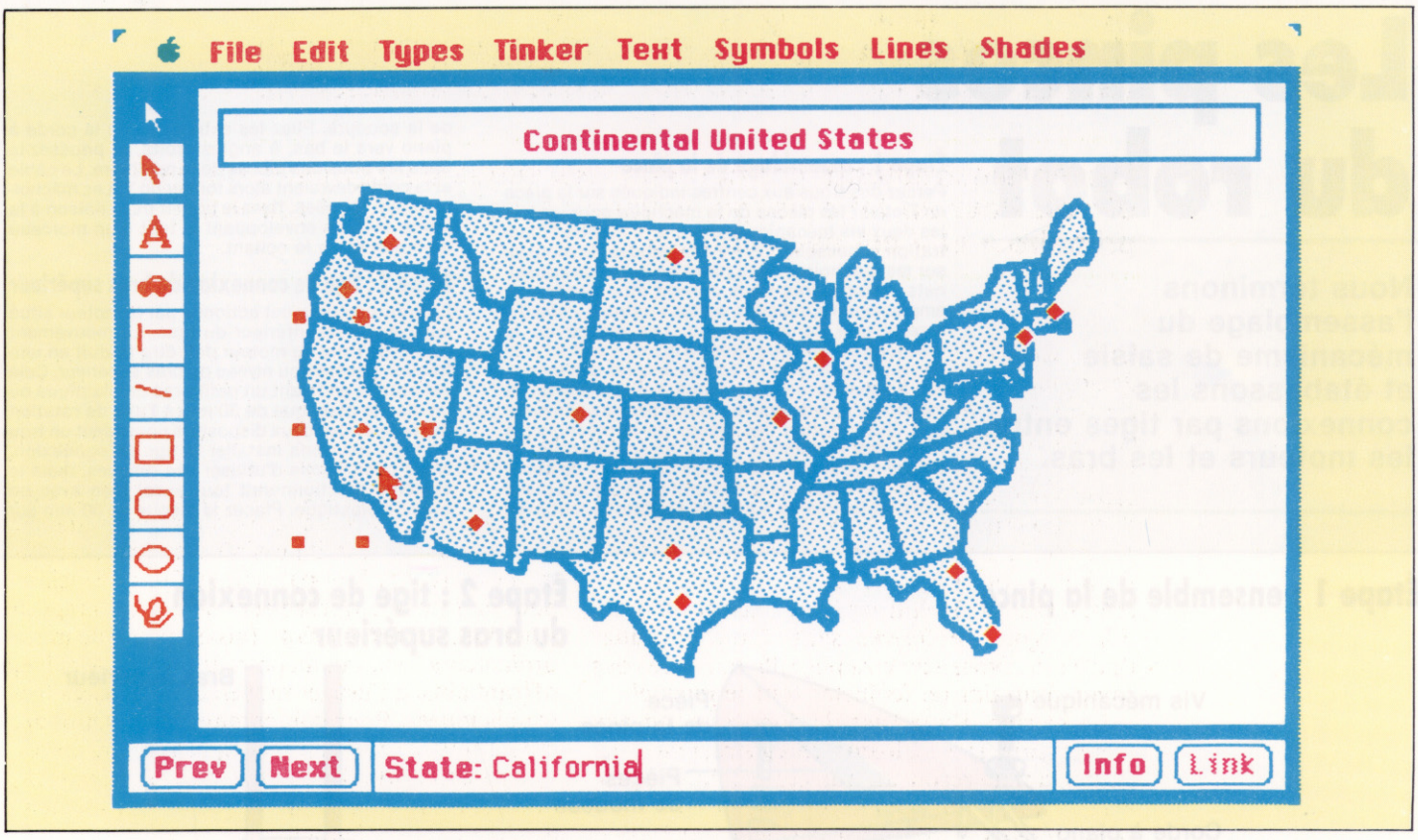
```
Carburateur
enlever la partie supérieure et régler le bouton

Réservoir d'essence
enlever le bouchon et remplir d'essence

Batterie
enlever le bouchon et remplir d'eau distillée

Lave-glace
enlever le bouchon et remplir d'eau
```

Lorsqu'Archive tourne sur le QL, l'écran est divisé en trois zones : au sommet de l'écran, tout ce qui concerne les commandes ; au milieu une zone de travail ; tandis que la partie inférieure est consacrée à l'affichage. Pour créer un nouveau fichier, il faut d'abord sélectionner la commande CREATE. Pour notre ensemble consacré à la mécanique automobile, nous taperons ainsi CREATE'VOITURE'<CR> (CR pour RETURN). Nous entrerons ensuite les noms de deux zones :

```
NOMDEPIÈCE$ <CR>
MISEENŒUVRE$ <CR>
<CR>
```

Le signe $ qui suit chaque zone signifie que la zone en question constitue une chaîne de caractères. <CR> à la fin des opérations marque l'achèvement de cette phase préliminaire.

Pour ajouter des enregistrements à la base de données ainsi créées, il faut alors taper la commande INSERT <CR>. Le nom de chacune des zones apparaît dans la mémoire de travail, ce qui vous permet d'y entrer les données nécessaires. On appuie sur F5 une fois terminé un enregistrement, qui est alors ajouté à la base de données.



Une fois que vous avez entré tous les éléments du fichier, vous pourrez quitter le mode INSERT en appuyant sur la touche Escape. Après quoi, vous taperez CLOSE pour que tout soit en ordre.

Un fichier créé avec Archive doit d'abord être ouvert (par la commande OPEN) avant que l'on puisse y faire des recherches. Cette commande permet à la fois la recherche et la modification des enregistrements, tandis que LOOK n'autorise qu'un simple examen. Dans les deux cas, il est possible de recourir à des instructions très simples : FIRST (pour voir le premier enregistrement du fichier), LAST (le dernier), NEXT (le suivant), BACK (le précédent).

Pour rechercher un enregistrement spécifique, il faut ajouter un argument à telle ou telle commande, ainsi FIND "CARBURATEUR". Le programme cherche alors à travers toutes les zones jusqu'à ce qu'il ait trouvé la chaîne en question. De plus, il est possible de préciser SEARCH NOMDEPIÈCE$ = "CARBURATEUR" ou MISEENŒUVRE$ = "RÉGLER". De telles précisions fonctionnent comme des opérateurs logiques, c'est-à-dire que la recherche continuera jusqu'à la découverte d'un enregistrement contenant la chaîne CARBURATEUR dans la zone NOMDEPIÈCE$ *et* d'une chaîne RÉGLER dans celle de MISEENŒUVRE$. Il est, de la même façon, tout à fait possible de faire usage de l'opérateur logique OR : un enregistrement sera sélectionné s'il contient la chaîne spécifiée dans une zone OU celle spécifiée dans une autre.

L'entrée des enregistrements dans une base de données est généralement effectuée au hasard, selon l'inspiration du moment. Il sera, en revanche, indispensable d'y accéder ou de les imprimer selon un ordre choisi.

Il est d'ailleurs possible de combiner un tri « primaire » avec un tri « secondaire ». Supposons que vous vouliez imprimer le contenu d'un fichier par ordre alphabétique des auteurs, chaque titre de chaque auteur étant également présenté de cette façon. Le résultat serait quelque chose de ce genre :

MOURLON J.-P.
Canards sauvages et d'élevage
Éditions du Col-Vert
0-85527-435-2
MOURLON J.-P.
Migrations atypiques en Europe du Nord
Éditions Défense de la Nature
0-7195-1332-4
MOURLON J.-P.
Trente-neuf recettes de canard
Éditions du Col-Vert
0-85527-438-6
MOURLON W.A.
Want to learn Japanese? It's easy as pie!
University of Bâton Rouge Press
226-52719-0

A supposer que les zones utilisées avec Archive aient pour nom AUTEUR$, TITRE$, ÉDITEUR$ et ISBN, il suffirait de taper une commande du type :

ORDER AUTEUR$; A, TITRE$, A

A signifie ici « classer par ordre alphabétique croissant ».

On voit qu'un logiciel de gestion de base de données se charge, sans difficultés, de tâches qui seraient très lourdes si elles étaient accomplies « à la main » : chercher les enregistrements significatifs, puis les classer selon certains critères.

Les États-Unis au doigt et à l'œil
Filevision fut d'abord créé pour les ordinateurs de type Apple II et III. Il fallut cependant attendre l'arrivée du Macintosh, ayant une puissance de traitement bien supérieure, pour qu'il puisse donner toute sa mesure. La souris permet de surcroît d'accéder à la base de données par simple déplacement d'un curseur. Un logiciel de création graphique très élaboré permet aussi la mise au point de graphismes, les formes et les symboles que vous tracez étant reliés à des champs spécifiques de la base de données. Notre illustration représente l'écran principal d'une base de données relatives aux États-Unis. Un simple déplacement de la souris permet de « pointer » la Californie et une pression sur un bouton nous donnera certaines informations sur cet État ; un nouveau « clic » ouvre une zone consacrée au texte, qui peut stocker beaucoup plus de renseignements. Toutes les données ainsi rassemblées peuvent être reliées ou triées de la même façon que dans toute base de données.

Les pinces du robot

Nous terminons l'assemblage du mécanisme de saisie et établissons les connexions par tiges entre les moteurs et les bras.

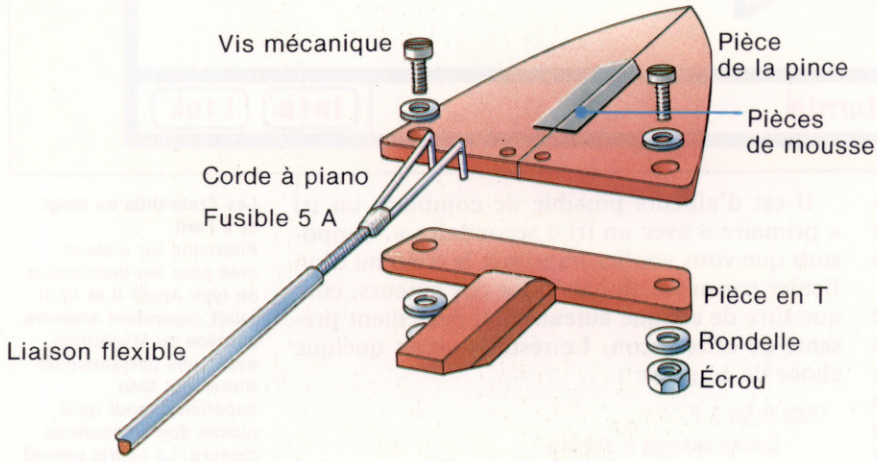
Étape 1 : assemblage de la pince

Percez des trous aux centres indiqués sur la pièce en T et sur les pièces de la mâchoire pour insérer les deux vis mécaniques comme le montre l'illustration ci-dessous. Collez des bandes de mousse sur les mâchoires de la pince en guise de coussinets. Deux autres trous doivent être percés aux emplacements indiqués sur les « patrons » afin d'y insérer une corde à piano. Les pièces de la mâchoire doivent être fixées à la pièce en T à l'aide de vis mécaniques et d'écrous. Insérez des rondelles au-dessus et au-dessous des pièces de la mâchoire, afin de pouvoir serrer les vis sans nuire à la libre rotation des pièces de la mâchoire. A l'une des extrémités de la liaison flexible de 500 mm, attachez deux bouts de 30 mm de corde à piano à l'extrémité du câble et de la corde à piano dans un fusible de 5 A et en scellant le tout avec de la soudure. Pliez les extrémités de la corde à piano vers le bas, à angles droits, et poussez-la dans les trous des pièces de la mâchoire. Le câble et la corde devraient alors former un Y. Les mâchoires étant fermées, fixez le boîtier de la liaison à la pince en T, en enveloppant le tout d'un morceau de coton et en le collant.

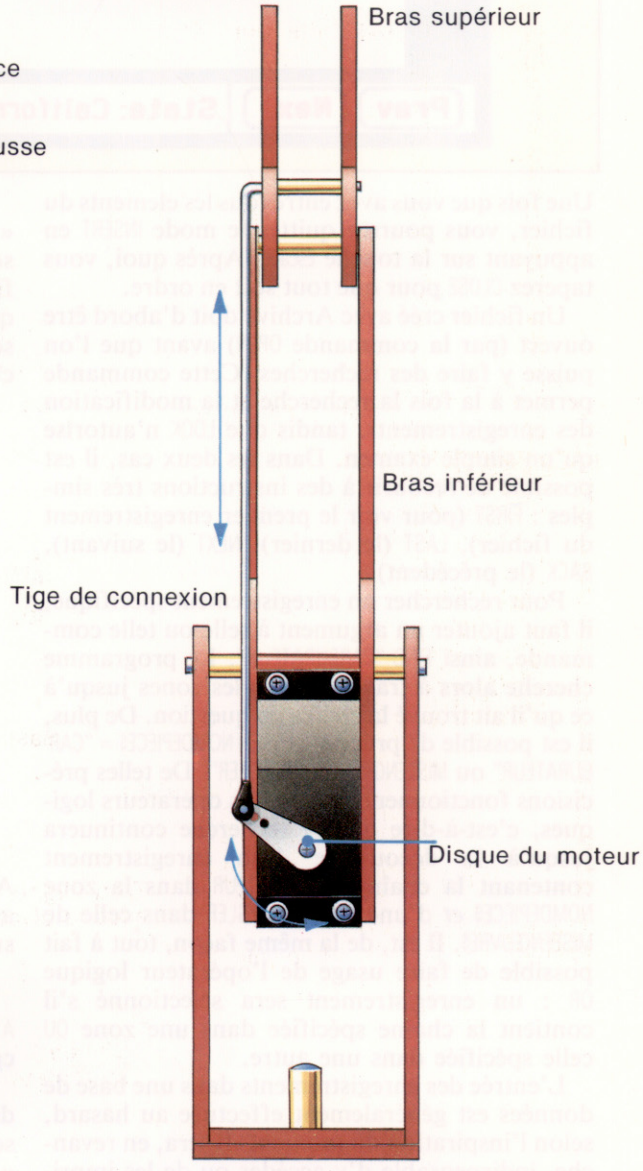
Étape 2 : tige de connexion du bras supérieur

Le bras supérieur est actionné par le moteur situé sur l'ensemble inférieur du bras. Le mouvement rotatif de l'axe du moteur doit être traduit en une traction-poussée au niveau du bras supérieur. Cela est obtenu en fixant un petit bras de plastique ou un disque plastique de 30 mm à l'axe de rotation du moteur. Ces deux dispositifs possèdent un trou où nous pourrons installer la tige de connexion. Nous choisissons d'utiliser des disques, mais le système fonctionnerait tout aussi bien avec un bras de plastique. Placez le disque de 30 mm sur

Étape 1 : ensemble de la pince

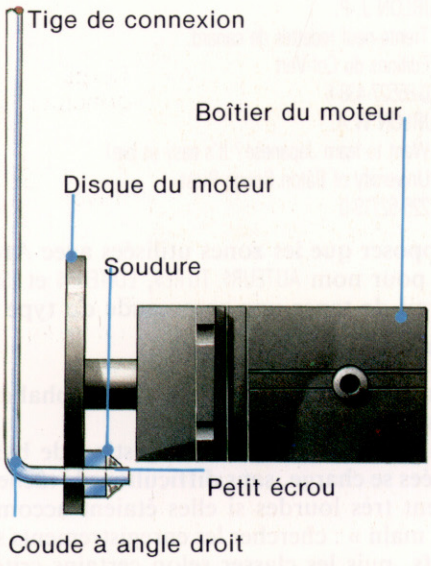


Étape 2 : tige de connexion du bras supérieur



Connexion des tiges aux disques

On peut trouver les tiges de connexion aux mêmes endroits que les servomoteurs. Sinon, une tige d'acier ordinaire de 2 mm peut être connectée au disque du moteur au moyen d'un coude de 90° fait près de l'extrémité de la tige. Il suffit d'introduire ce coude dans le trou du disque et de souder un petit écrou à son dos. Cette opération est facilitée en retirant le disque de l'axe du moteur.
Aux endroits où les tiges pénètrent dans les membres en bois du bras, il est recommandé de garnir l'intérieur du trou d'un petit bout de cuivre dont le diamètre interne est tout juste plus grand que la tige.



Kevin Jones

l'axe de rotation de façon que, lorsqu'on tourne le moteur à fond dans le sens inverse des aiguilles d'une montre, le trou se trouve tout juste à la gauche de la position « 6 heures » (en imaginant que le disque du moteur est une plaque d'horloge où la position « 12 heures » se trouve tout en haut). Cette position est facile à obtenir.

Maintenant, prenez une tige de 2 mm de diamètre et pliez des coudes à angle droit aux deux extrémités, afin de pouvoir insérer la tige dans le trou du disque du moteur et dans les deux trous préparés à cet effet sur les pièces supérieures du bras, près du joint du coude. La longueur de la tige doit être telle que les bras inférieur et supérieur forment un angle de 90° au niveau du coude, lorsque le moteur est tourné à fond dans le sens antihoraire. Vérifiez l'action du bras en tournant lentement le moteur dans le sens horaire, manuellement. Lorsque l'action traction-poussée de la tige est satisfaisante, collez ou soudez la tige sur le disque du moteur.

Étape 3 : tige de connexion du bras inférieur

Montez les deux autres moteurs sur le corps principal. Le moteur de droite actionne le bras inférieur. Faites tourner le moteur à fond dans le sens inverse des aiguilles (vu du côté disque du moteur) et positionnez le disque de telle sorte qu'il se trouve juste à droite de la position « 6 heures ». Fixez une tige d'acier de 2 mm entre le disque du moteur et les trous du bras inférieur près du joint de l'épaule, comme précédemment. La longueur de la tige doit être telle que le bras inférieur soit approximativement horizontal par rapport au moteur tourné complètement dans le sens inverse des aiguilles d'une montre. De nouveau, vérifiez l'action du bras inférieur en faisant tourner le moteur manuellement, puis collez la tige de connexion au disque du moteur lorsque le tout est satisfaisant.

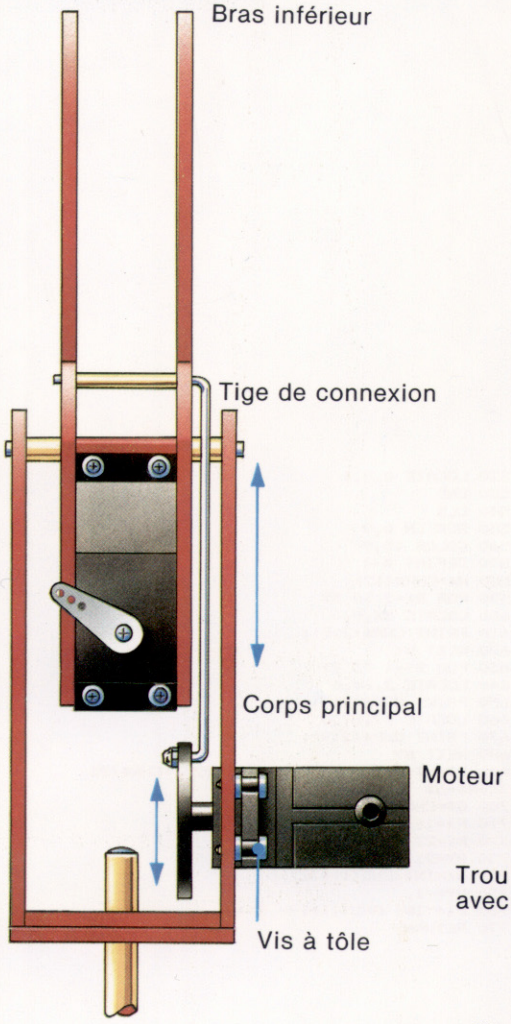
Étape 4 : connexion de la liaison flexible du moteur

Le moteur de gauche monté sur le corps principal sert à ouvrir et à fermer la pince en poussant et en tirant le câble dans la liaison flexible. Commencez par percer un trou dans la pièce transversale du corps principal, afin de pouvoir y insérer le câble par derrière. Collez le boîtier externe de la liaison dans ce trou. Une fois la colle sèche, tirez le câble à travers ce trou pour fermer à fond la mâchoire de la pince et fixez l'extrémité du câble au disque du moteur. Le trou sur la surface du disque doit se trouver à la position « 9 heures » lorsque le moteur est tourné à fond dans le sens inverse des aiguilles. L'extrémité du câble peut être fixée de plusieurs manières au disque du moteur. Il est préférable de monter un petit serre-câble sur le disque. Sinon, on peut également plier le câble à 90° et enfiler l'extrémité du câble dans le trou pour l'y coller ou l'y souder. Peu importe la manière dont vous vous y prenez, il est important que le moteur puisse tirer et pousser le câble librement pour ouvrir et fermer la pince.

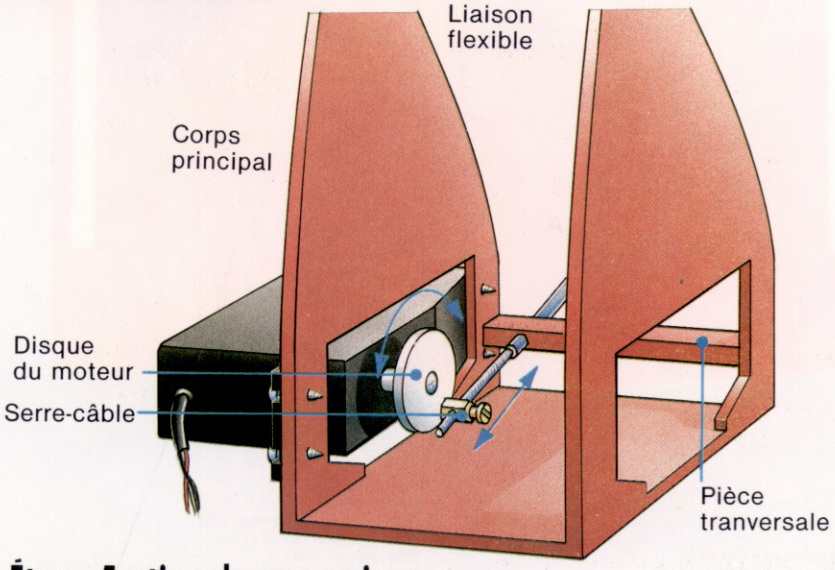
Étape 5 : tige de connexion du corps principal

La connexion finale à effectuer est celle qui relie le moteur monté dans la base au corps principal. Percez d'abord un petit trou dans la base du corps principal (vous devrez retirer l'ensemble hors de la cheville d'acier pour le faire). Le trou doit être aligné avec le pivot, près du bord droit de la base (vue du dessus) du corps principal. Après avoir percé le trou, remettez l'ensemble du corps principal sur la cheville et mettez-le à la position « 2 heures ». Tournez le moteur à fond dans le sens inverse des aiguilles d'une montre et positionnez le disque de telle sorte qu'il se trouve juste à la droite de la position « 6 heures ». Fixez une tige de connexion comme précédemment et vérifiez la libre rotation dans le sens inverse des aiguilles d'une montre.

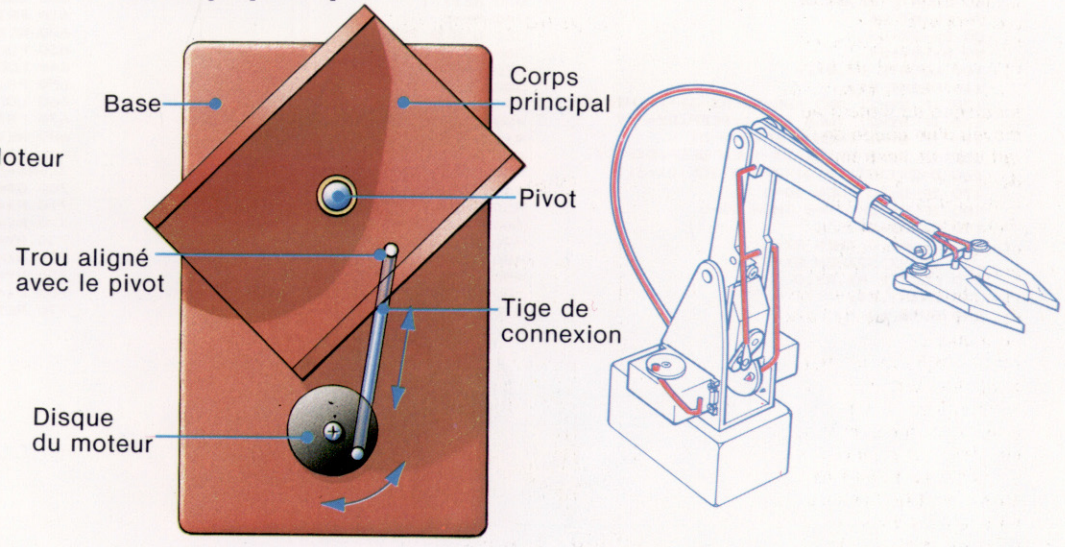
Étape 3 : tige de connexion du bras inférieur



Étape 4 : connexion de la liaison flexible au moteur

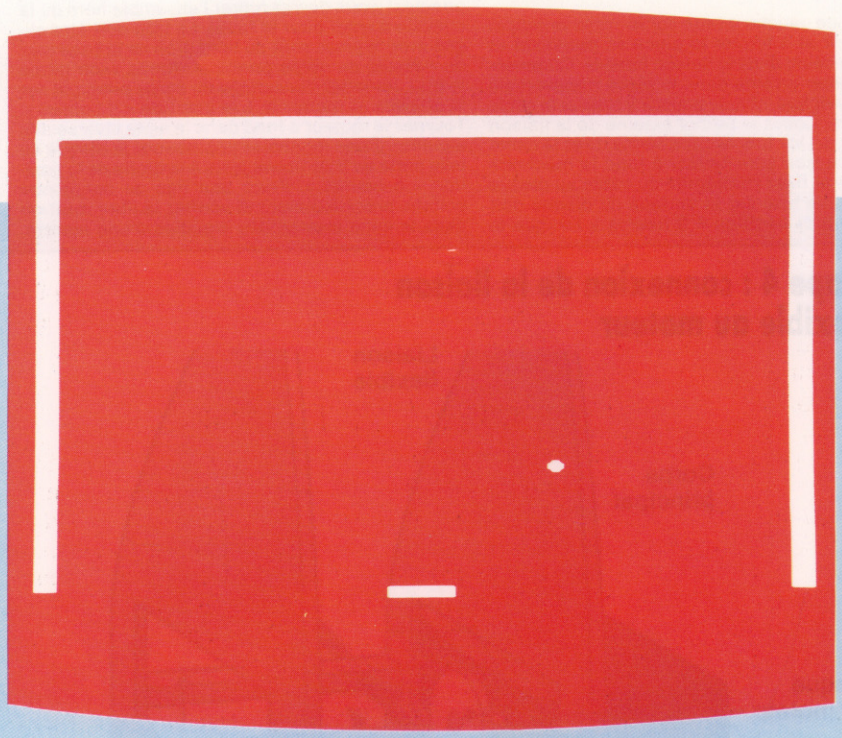


Étape 5 : tige de connexion du corps principal



Squash en MSX

Les jeux de Squash existent pour de nombreuses versions de micro-ordinateurs. Nous vous avons déjà donné plusieurs programmes. Voici le listage pour micros MSX.



Grâce à votre ordinateur, vous pouvez jouer au squash assis dans un fauteuil. La raquette se déplace à l'aide des touches du curseur. Vous disposez de dix balles que vous devez garder en jeu le plus longtemps possible. Chaque balle renvoyée rapporte un point.

```
10 REM **********
20 REM * SQUASH *
30 REM **********
40 KEY OFF
50 GOSUB 540
60 LOCATE BX,BY,0
70 PRINT N$
80 BX=BX+DX
90 BY=BY+DY
100 LOCATE BX,BY,0
110 PRINT B$;
120 IF BY=21 AND ABS(BX-RX-3)>1 THEN 250
130 IF BY=21 THEN S=S+1:BEEP:DY=-DY
140 IF BY=1 THEN BEEP:DY=-DY
150 IF BX=3 OR BX=35 THEN BEEP:DX=-DX
160 D=2*((STICK(0)=7)-(STICK(0)=3))
170 IF D<>0 THEN D0=D
180 IF STICK(0)=0 THEN D0=0
190 RX=RX+D0
200 IF RX<0 THEN RX=0
210 IF RX>32THEN RX=32
220 LOCATE RX,RY,0
230 PRINT R$;
240 GOTO 60
250 NB=NB+1
260 IF NB=11 THEN 370
270 LOCATE BX,BY,0
280 PRINT N$;
290 FOR I=1 TO 3
300 BEEP
310 FOR J=1 TO 100
320 NEXT J
330 NEXT I
340 D0=0
350 GOSUB 720
360 GOTO 60
370 LOCATE 13,5,0
380 PRINT "SCORE :";S;
390 IF S>R1 THEN R1=S
400 LOCATE 13,10,0
410 PRINT "RECORD :";R1;
420 LOCATE 13,15
430 PRINT "UNE AUTRE ?";
440 NB=0
450 S=0
460 D$=INKEY$
470 IF D$<>"" THEN 460
480 D$=INKEY$
490 IF D$="" THEN 480
500 IF D$<>"N" AND D$<>"n" THEN 50
510 CLS
520 LOCATE 0,0,1
530 END
540 CLS
550 SCREEN 0,0
560 COLOR 15,9
570 DEFINT A-Z
580 N$=CHR$(32)
590 FOR BX=2 TO 37
600 LOCATE BX,0,0
610 PRINT CHR$(219);
620 NEXT BX
630 FOR BY=1 TO 21
640 LOCATE 2,BY,0
650 PRINT CHR$(219);
660 LOCATE 37,BY,0
670 PRINT CHR$(219);
680 NEXT BY
690 R$=N$+N$+CHR$(223)+CHR$(223)+CHR$(22
3)+N$+N$
700 B$=CHR$(249)
710 RX=16
720 RY=22
730 BY=21
740 BX=INT(RND(1)*32)+5
750 DY=-1
760 DX=(INT(RND(1)*2)-.5)*2
770 RETURN
```

Tranche de mémoire

L'arrivée du système microlecteur Sinclair n'a pas empêché des sociétés indépendantes de développer pour le Spectrum d'autres systèmes de stockage. Parmi ceux-ci, le Wafadrive de Rotronics.

Le fait que le Spectrum soit un micro de grande diffusion n'a pas convaincu pour autant certains milieux qui le considèrent toujours comme inapproprié à un usage professionnel. Au point qu'il fut relégué au rang d'une machine à jeux. Le problème vient en partie du clavier qui ne permet pas l'utilisation confortable de logiciels tels que les traitements de texte et les bases de données. Sinclair Research a dernièrement essayé d'y remédier en lançant le Spectrum + doté d'un clavier semblable à celui du QL.

Ce n'est pourtant qu'un aspect du problème : le manque d'interfaces standards et, plus important encore, l'absence d'un système performant de mémoire de masse sont autant de handicaps sérieux. Le manque de sauvegarde exclut pratiquement toute utilisation professionnelle, mais également toute activité personnelle digne de ce nom. L'introduction de l'Interface 1 et du Microdrive remédie à ces défaillances. Néanmoins, le doute a subsisté quant à l'efficacité du Microdrive. En outre, s'il y a énormément de logiciels sur cassettes pour le Spectrum, il en existe très peu qui soient adaptés au Microdrive. Lorsque ce genre de situation se fait jour, des fabricants indépendants peuvent intervenir pour proposer des alternatives. Nous voyons aujourd'hui l'un des deux rivaux dans la course à la fourniture d'une mémoire de masse au Spectrum : le Wafadrive de Rotronics ; l'autre étant le Discovery 1 de Opus Supplies.

Contrairement à l'Interface 1 et aux unités microlecteurs qui l'accompagnent, le Wafadrive est en un seul bloc. Les interfaces périphériques et les unités de sauvegarde sont contenues dans un seul boîtier. L'avantage est l'absence des câbles de liaison nécessaires au système Sinclair. En revanche, le Wafadrive n'a pas la souplesse de configuration des Microdrive, qui peuvent être chaînés en étoile pour accroître la mémoire de masse.

De la place à l'arrière
Contrairement au système Sinclair de mémoire de masse destiné au Spectrum, le Wafadrive de Rotronics est tout d'un bloc.
Il contient un lecteur double, un port RS232 et une interface Centronics. Le câble-nappe se loge dans le connecteur arrière du Spectrum. Ainsi situé, les lecteurs se trouvent au-dessus du clavier et sont facilement accessibles.
(Cl. Chris Stevens.)

L'aspect de la machine

Le Wafadrive est un boîtier noir doté d'un câble-ruban à trente-cinq voies se connectant dans la case à cartouche située sur la carte Bus d'extension du Spectrum. L'avant du Wafadrive dispose d'un double microlecteur. Entre les deux lecteurs, trois diodes électroluminescentes signalent la mise sous tension (au centre) et le fonctionnement des lecteurs (à droite et à gauche).

L'arrière du Wafadrive comporte trois connecteurs latéraux. Sur la gauche, une carte Bus d'extension parallèle pour la liaison avec l'Interface 2. Au centre, l'interface compatible Centronics pour une sortie parallèle vers une imprimante. A droite, un port série RS232 permettant l'interfaçage avec des modems et autres périphériques série. Ces interfaces constituent une réelle amélioration sur ceux de l'Interface 1 de Sinclair qui supposait, par exemple, la connexion d'une seconde interface Centronics sur le connecteur latéral pour la sortie parallèle imprimante. Les utilisateurs auront néanmoins de la difficulté à trouver des imprimantes Centronics ou des modems dotés de connecteurs à cartouche compatibles microlecteurs.

Les microlecteurs conçus spécialement pour le Wafadrive sont très semblables à ceux utilisés par le Sinclair. Chacun contient une bande sans fin de type vidéocassette d'une largeur de 1,8 mm. Cette bande est préférée à la bande conventionnelle de type audio, du fait de sa plus grande endurance et de ses grandes ressources en sauvegarde d'informations. Une fois formatée, la bande peut contenir jusqu'à 128 K, Rotronics offrant également des cartouches de 64 et 16 K.

Les cartouches elles-mêmes sont deux fois plus larges que celles de Sinclair. Une fois dans leur boîte, elles ont cependant pratiquement les mêmes longueur et largeur. Cela donne aux cartouches Rotronics l'aspect de cassettes miniatures. Les

cartouches du Wafadrive n'ont pas besoin de boîtier de protection; la fragile bande est en effet protégée par un couvercle se rabattant automatiquement. Ce système est semblable aux microlecteurs Sony 3 ½. La protection des Rotronics est en plastique et non en métal comme pour les Sony. Le côté gauche de la cartouche comporte un taquet de protection en écriture qu'il est possible de retirer. Ce taquet ne pouvant être remis en place une fois arraché, les utilisateurs devront trouver un autre moyen pour protéger leur cartouche (un petit autocollant par exemple).

Les commandes utilisées par le Wafadrive sont plus ou moins identiques à celles utilisées par les microlecteurs Sinclair. Les commandes doivent être suivies, sur les deux systèmes, du signe * demandant l'accès du périphérique externe de sauvegarde. Comme exemples de cette syntaxe, on peut donner SAVE *, LOAD *, et VERIFY *. Le système Wafadrive comporte néanmoins quelques particularités, du fait de la présence de deux microlecteurs, par opposition aux nombreux microlecteurs susceptibles d'être en batterie avec d'autres systèmes. Ainsi, au formatage d'une cartouche Sinclair, la commande FORMAT 'm'; 0; «nom» utilise 'm'; 0 pour désigner le numéro du microlecteur utilisé. Avec le Wafadrive, la commande devient FORMAT * 'a:nom', «a:» donnant le nom du lecteur. Vous remarquerez qu'avec le Wafadrive le choix est entre les lecteurs « a » et « b » seulement, le Microdrive autorisant un choix compris entre zéro et sept.

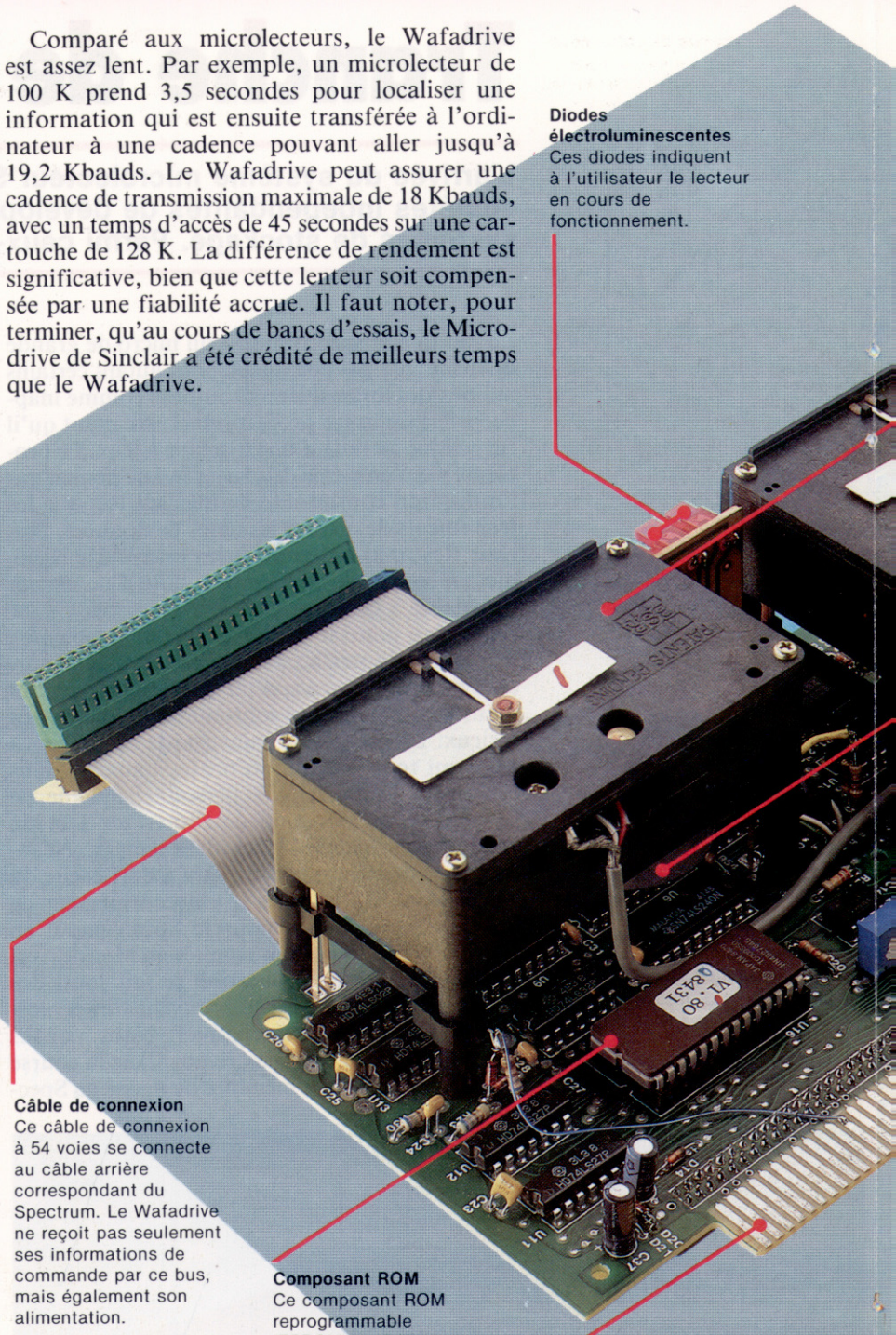
Le système des canaux

Le Wafadrive utilise en outre le système dit « de canaux » du Spectrum. Ce dernier consiste en seize canaux réservés au traitement des entrées/sorties. Certains sont destinés à l'écran et à l'imprimante. Les canaux 4 et 16 sont pour les autres périphériques. Les canaux de sortie vers le Wafadrive sont mobilisés par la commande OPEN #. Le Wafadrive ajoute deux autres canaux au système. Les canaux « r » et « c » (lettres que l'on peut aussi écrire en majuscules) sont réservés respectivement aux interfaces RS232 et Centronics. Leur utilisation est semblable à celle des canaux « t » et « b » pour l'accès du port RS232 sur l'Interface 1.

Une mémoire morte de 8 K est incorporée pour les commandes du BASIC étendu utilisées pour la gestion du système. Le système d'exploitation du Wafadrive (WOS/Wafadrive Operating System) peut fonctionner selon le mode de remplissage dit « paging out » de la ROM. Les 8 K les plus bas sont ainsi remplis à partir de la mémoire morte du Spectrum, d'une manière très similaire à celle employée par l'Interface 1.

Pour prendre un exemple, la commande LOAD * génère une erreur sur le Spectrum; aussi, lorsque l'interpréteur lit cette commande à l'écran, il appelle la routine BASIC de gestion d'erreur. Néanmoins, cette commande d'appel sera interceptée par le WOS qui chargera la ROM du Wafadrive. Cela aura pour effet l'interprétation de LOAD * comme une commande valide.

Comparé aux microlecteurs, le Wafadrive est assez lent. Par exemple, un microlecteur de 100 K prend 3,5 secondes pour localiser une information qui est ensuite transférée à l'ordinateur à une cadence pouvant aller jusqu'à 19,2 Kbauds. Le Wafadrive peut assurer une cadence de transmission maximale de 18 Kbauds, avec un temps d'accès de 45 secondes sur une cartouche de 128 K. La différence de rendement est significative, bien que cette lenteur soit compensée par une fiabilité accrue. Il faut noter, pour terminer, qu'au cours de bancs d'essais, le Microdrive de Sinclair a été crédité de meilleurs temps que le Wafadrive.



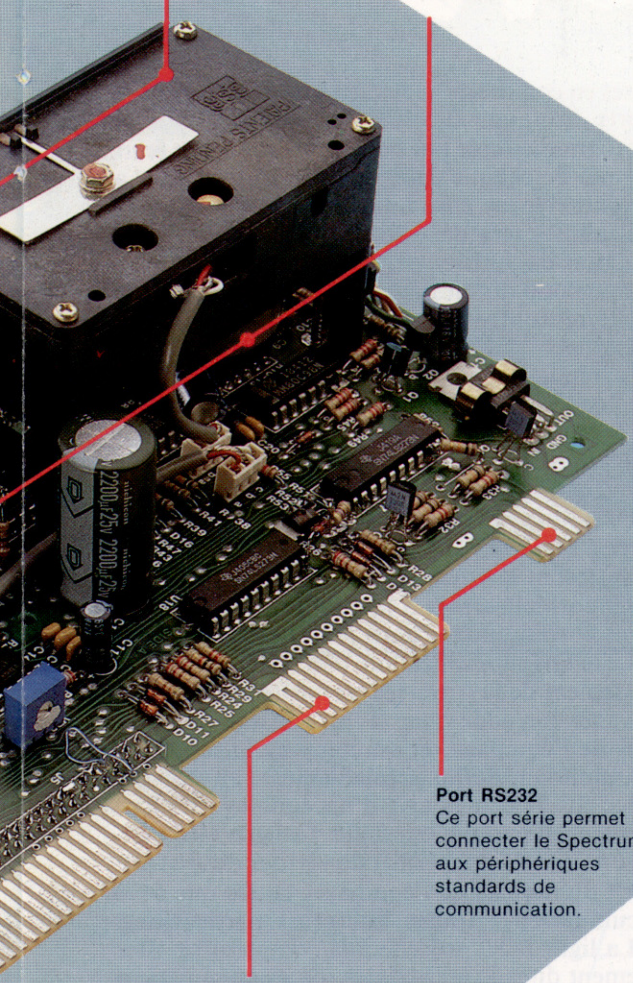
Diodes électroluminescentes
Ces diodes indiquent à l'utilisateur le lecteur en cours de fonctionnement.

Câble de connexion
Ce câble de connexion à 54 voies se connecte au câble arrière correspondant du Spectrum. Le Wafadrive ne reçoit pas seulement ses informations de commande par ce bus, mais également son alimentation.

Composant ROM
Ce composant ROM reprogrammable (EPROM) de 8 K contient le système d'exploitation du Wafadrive (WOS).

Connexion sortie parallèle
Cette connexion latérale autorise l'adjonction d'interfaces supplémentaires compatibles Spectrum.

Bien que ces temps d'accès soient beaucoup plus rapides que ceux des cassettes, ils restent très lents par rapport à ceux des lecteurs de disques. Le Wafadrive, tout comme le Microdrive, comporte une procédure très commode lors de l'accès au catalogue de la bande. Le catalogue figure sur le premier secteur de la cartouche, juste après le point de collage des deux extrémités de la bande (bande sans fin). Cette particularité permet au système de savoir facilement repérer le catalogue : le lecteur passe la bande en revue jusqu'à rencontrer sa fin, et le premier secteur immédiatement après le contient. Après quelques



Lecteurs de cartouches
Le Wafadrive comporte deux lecteurs de bandes autonomes. Rotronics propose des cartouches correspondantes de 16, 64 et 128 K.

Moteurs
Chaque lecteur a son propre moteur électrique destiné au défilement de la bande dans la cartouche.

Port RS232
Ce port série permet de connecter le Spectrum aux périphériques standards de communication.

Interface Centronics
Le Wafadrive est équipé d'un port Centronics en vue d'une liaison imprimante — indispensable pour le traitement de texte.

secondes, la tête du lecteur aura dépassé le secteur du catalogue. Si l'on demande alors à nouveau un catalogue, au lieu d'embobiner à nouveau toute la bande, le lecteur n'attendra pas une seconde avant de l'afficher. La raison en est que le catalogue, une fois appelé, figure en RAM. Le système d'exploitation se contente alors d'examiner le secteur suivant pour vérifier qu'il s'agit bien de la même cartouche. Dans l'affirmative, le WOS affiche le catalogue présent dans sa RAM.

En ayant présent à l'esprit l'idée de mettre en œuvre un système de mémoire de masse pour le Spectrum afin de le rendre crédible, Rotronics a inclu dans sa cartouche le programme de traitement de texte Spectral Writer. Il s'agit d'un système relativement complet qui tire pleinement parti du Wafadrive. Des fonctions telles que la refonte de paragraphes, l'insertion de mots ou la suppression de lignes entières sont appelées par la touche « Bascule » du Spectrum, en conjonction avec d'autres touches du clavier. Les fonctions d'accès aux fichiers sur cartouche sont obtenues par les commandes d'options; parmi celles-ci : SAVE et LOAD (respectivement sauvegarde et chargement des fichiers depuis cassette ou Wafadrive). Spectral Writer est un bon traitement de texte, même s'il ne vous permet pas de fixer la longueur de ligne à l'écran. Il est néanmoins regrettable que, même en utilisant le Spectrum +, la qualité médiocre du clavier compromette l'efficacité de Spectral Writer.

Ce qui permet à un périphérique de mémoire de masse de s'affirmer, c'est, en fin de compte, l'appui des concepteurs de logiciels. Cela ne semble présentement pas acquis pour le Wafadrive. En effet, aucun des principaux éditeurs de logiciels n'a encore produit de programmes sur cartouches Wafadrive (Sinclair a, du reste, rencontré le même problème). Tout n'est cependant pas joué. Il existe, en effet, une société qui a produit un programme permettant de recopier sur cartouche Wafadrive tout programme commercial. Cela suppose évidemment d'acheter la cassette du programme et une cartouche pour la transférer; mais c'est une petite dépense par rapport aux gains énormes en temps d'accès qu'elle permet.

Une autre difficulté mineure avec le Wafadrive tient à ses connecteurs-nappes à l'arrière de la machine. Ils ne sont pas standards, ce qui suppose que les utilisateurs transforment eux-mêmes les interfaces, faute de quoi ils auront beaucoup de mal à trouver des périphériques.

ROTRONICS WAFADRIVE

DIMENSIONS
230 × 110 × 80 mm.

INTERFACES
Port série RS232, interfaces parallèles Centronics, connecteur latéral Spectrum.

FORMAT
Lecteurs de disques souples en boucle sans fin.

CAPACITÉ
Cartouches 16, 64 et 128 K.

VITESSE
Vitesse de transfert 16 Kbauds, temps maximal d'accès 6,5 s (16 K), 4,5 s (128 K).

Cartouche pour Wafadrive
La cartouche du Wafadrive a environ deux fois la taille de celle du Microdrive. Elle ressemble à une cassette, mais la bande à l'intérieur est sans fin. Cela signifie qu'elle ne doit pas être rebobinée pour permettre l'accès à des données que la tête de lecture/écriture a dépassé. (Cl. Chris Stevens.)

Haute mer

Nous voici maintenant en route pour notre grand voyage vers le Nouveau Monde. Reste à tenir un relevé hebdomadaire de l'état du navire et de l'équipage.

Pendant notre voyage, l'allure à laquelle avance le bateau dépend aussi de l'état de santé de l'équipage : si celui-ci est en parfaite santé, il n'ira certes pas plus vite, mais des matelots très malades le ralentiront ; et le jeu prendra fin si, bien entendu, ils passent tous de vie à trépas.

L'une des fonctions du journal de bord tenu par le capitaine est d'informer le joueur de la mort d'un membre de l'équipage, de l'épuisement de telle ou telle provision, ainsi que de donner une nouvelle estimation de la durée du voyage compte tenu de ces nouveaux facteurs. Une nouvelle variable, EW, est créée ligne 41. Elle représente le temps supplémentaire (exprimé en semaines ou en fractions de semaine) qui viendra s'ajouter aux huit semaines prévues à l'origine, au cas où la force additionnée de tous les marins tomberait en dessous d'un certain niveau. Cette durée est calculée, stockée en EW sous forme d'un nombre réel, puis ajoutée à JL sous forme d'un entier (à l'aide de INT), de façon que JL puisse être affiché et incrémenté à l'aide de nombres entiers sans perdre la partie fractionnelle de EW. Le sous-programme consacré au relevé de fin de semaine commence ligne 5300. Une bonne partie du programme traitera des aléas de la traversée ; ce sera le rôle de nouveaux modules. Tous ces événements affecteront l'équipage, les provisions, et bien sûr la durée du trajet : ils influeront donc fortement sur les variables mises en œuvre ici. Le sous-programme définit d'abord, ligne 5325, une boucle qui vérifie si un membre de l'équipage est mort ou non. Il faut, pour cela, lire à la suite le second élément du tableau affecté aux matelots, TS(,), pour chacun des seize hommes ; si la force de l'un d'eux est de – 999, cela indique qu'il est décédé.

Dans ce cas, le malheureux est ajouté à X (compte hebdomadaire des victimes). Sinon, la boucle est parcourue une nouvelle fois. En cas de décès, la ligne 5340 remet à zéro les éléments correspondant à la catégorie de matelot et à la force, afin d'empêcher qu'ils ne soient pris en compte lors du prochain relevé hebdomadaire. Enfin, le matelot est soustrait du total des hommes d'équipage, CN. Si personne n'est mort (X reste alors à zéro à la fin de la boucle), le programme passe ligne 5400 et commence à vérifier l'état des provisions. Il se peut, malheureusement pour vous, que tous vos marins aient péri. Cela provoque l'affichage du message : 'tous les survivants ont à leur tour péri. Votre navire erre à l'aventure sur l'océan indifférent. Fin de la partie.' Les provisions déjà consommées sont prises en compte de la même façon que les marins morts. Pour voir si certaines sont déjà épuisées, le programme crée une boucle de 1 à 4 (pour tenir compte de chaque catégorie). Si l'une d'elles a été entièrement consommée, sa valeur dans le tableau PA() sera de – 999. La ligne 5415 a pour fonction de vérifier ce tableau. Elle provoque l'impression du message 'vous êtes désormais à court de...' si elle rencontre cette valeur particulière.

Là encore, l'élément correspondant de PA() sera remis à zéro, toujours pour éviter des confusions la semaine suivante. Il y a alors diverses possibilités. Par exemple, plusieurs provisions peuvent être épuisées. Dans ce cas, le programme donne leur nom et précise ET DE. Il se peut aussi que l'une d'elles puisse ultérieurement être récupérée à l'occasion d'un des nombreux hasards qui marqueront le voyage ; la situation redeviendra alors normale, grâce à la modification de l'élément correspondant dans le tableau PA().

Le temps supplémentaire nécessaire à l'accomplissement de la traversée est calculé de la façon suivante : on additionne la force de tous les membres de l'équipage, et on compare le résultat à un minimum défini préalablement, en l'occurrence 800. Il peut s'agir d'un équipage de huit hommes en parfaite santé, etc. La ligne 5465 définit une boucle qui permet de vérifier chaque élément du tableau de l'équipage, tandis que 5470 accumule en X toutes les forces particulières.

La ligne 5480 procède à la comparaison proprement dite. Si le total des forces est supérieur à 799, la durée totale du voyage ne connaîtra aucune augmentation. Le programme passera alors ligne 5494, qui vous demandera d'appuyer sur une touche pour passer à la semaine suivante.

Si, en revanche, le total est inférieur à 800, la formule de la ligne 5489 permet de calculer la durée supplémentaire. Elle soustrait de 800 la force totale de l'équipage, divise le résultat par 800 et ajoute le tout à EW, sous forme d'un nombre réel. La ligne 5490 ajoute à la durée totale du voyage la partie entière de EW, de façon à éviter que le chiffre ne soit pas affiché sous forme fractionnaire. EW est, sous forme cumulée, ajoutée à JL à la fin de chaque semaine. Il faudra insérer la ligne suivante :

```
9315 IF TS(S1,2)=0 THEN 9340
```

Cela évitera que la force d'un membre d'équipage décédé ne soit diminuée de WF dans le sous-programme (ligne 9300) donné précédemment.

Module cinq : journal de bord

Rapport de fin de semaine

```
5300 REM RELEVE FIN DE SEMAINE
5305 PRINTCHR$(147)
5310 S$="          JOURNAL DE BORD*":GOSUB9100
5312 S$="                          *":GOSUB9100
5314 GOSUB9200
5316 PRINT:PRINT"       FIN DE SEMAINE:WK
5318 GOSUB9200
5320 X=0
5325 FORT=1TO16
5330 IFTS(T,2)<>-999THEN5350
5335 X=X+1
5340 TS(T,1)=0:TS(T,2)=0
5345 CN=CN-1
5350 NEXT
5355 IFX=0THEN5400
5358 PRINT:PRINT
5360 S$="LA SEMAINE PASSEE*":GOSUB9100
5365 IFCN)0THEN5390
5367 S$="TOUS LES SURVIVANTS ONT A LEUR TOUR PERI*":GOSUB9100
5369 GOSUB9200:PRINT:GOSUB9200
5373 S$="VOTRE NAVIRE ERRE A L'AVENTURE*":GOSUB9100
5374 GOSUB9200
5376 S$="SUR L'OCEAN INDIFFERENT...*":GOSUB9100
5378 GOSUB9200
5380 PRINT:PRINT
5382 S$="     FIN DE LA PARTIE*":GOSUB9100
5384 PRINT:PRINT
5386 GOSUB9200:END
5388 GOTO5386
5390 PRINTX:
5392 IF X=1 THEN S$="MEMBRE D'EQUIPAGE EST MORT*"
5394 IFX<>1 THEN S$="MEMBRES DE L'EQUIPAGE SONT MORTS*"
5396 GOSUB9100
5398 GOSUB9200
5400 PRINT:PRINT
5405 S$="VOUS ETES A COURT DE"
5410 FOR T=1TO4
5415 IFPA(T)<>-999THEN5440
5420 PRINTS$:P$(T)
5425 PA(T)=0
5428 S$="                ET DE"
5440 NEXT
5450 GOSUB9200
5455 REM CALCULE VALEUR JL
5460 X=0
5465 FORT=1TO16
5470 X=X+TS(T,2)
5475 NEXT
5480 IFX>799THEN5494
5481 PRINT:PRINT
5482 S$="L'EQUIPAGE EST EN-DESSOUS DE LA FORCE NECESSAIRE*":
GOSUB9100
5483 S$="ET LE VOYAGE PEUT ETRE PLUS LONG*":GOSUB9100
5489 EW=EW+(800-X)/800)
5490 JL=JL+INT(EW)
5492 PRINT
5494 S$=K$:GOSUB9100
5496 GETI$:IFI$=""THEN5496
5499 RETURN
```

Variable semaines supplémentaires

```
41 EW=0: REM SEMAINES EN PLUS
```

Addition à la boucle principale

```
880 GOSUB 5300: REM RELEVE FIN DE SEMAINE
```

Variantes de basic

Spectrum :

Procédez aux modifications suivantes :

```
5305 CLS
5496 LET I$=INKEY$=«»THEN GOTO 5496
```

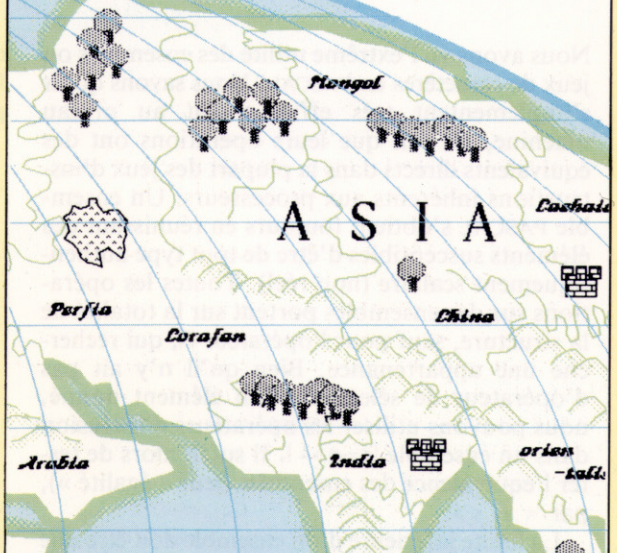
BBC Micro :

Procédez aux modifications suivantes :

```
5305 CLS
5496 I$=GET$
```

Ian McKinnell

Où en sommes-nous ?
Le cinquième module de notre programme vient d'être présenté. Nous pouvons dès maintenant passer en revue les premiers stades de notre jeu de simulation grâce à un organigramme. Au début : embauche, achat des marchandises et des provisions.



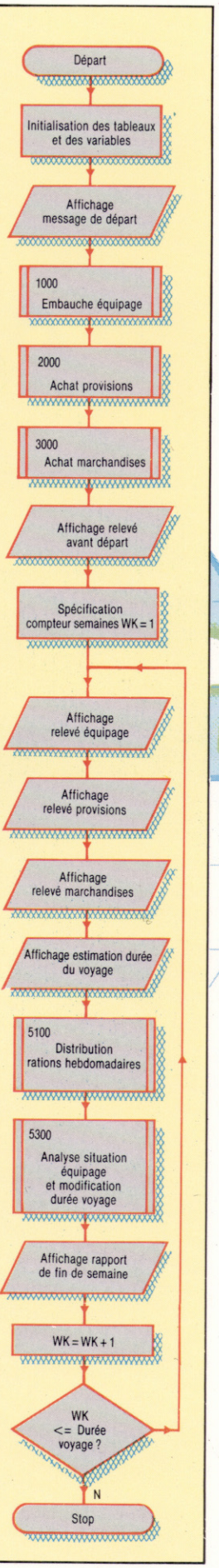
Il peut être intéressant, à ce stade du jeu, d'étudier les effets d'une réduction des forces individuelles, en insérant un petit sous-programme qui accorde à celles-ci, en début de voyage, des valeurs aléatoires. Si, toutefois, vous décidez de faire usage des lignes 601 à 604, n'oubliez pas de les faire disparaître avant de taper le module qui sera présenté ultérieurement :

```
601 REM TEST
602 FOR T=1 TO 16
603 IF TS(T,2)=100 THEN TS(T,2)=
    INT(RND(1)*100)+1
604 NEXT T
```

Le sous-programme commence, ligne 603, par parcourir seize fois de suite le second élément du tableau consacré à l'équipage. Toute force fixée à 100 reçoit alors une valeur aléatoire comprise entre 0 et 99, grâce à l'emploi de la formule RND ; on ajoute 1 à chaque résultat obtenu, de façon que, à aucun moment, la force de tel ou tel ne tombe à zéro, ce qui lui vaudrait une mort immédiate.

Le relevé de fin de semaine est appelé au sein de la boucle principale (lignes 820 à 899) par un GOSUB. Cette boucle était à l'origine contrôlée par un FOR...NEXT dont la limite supérieure était déterminée en fonction de JL, qui fixe la durée du voyage. Mais le nouveau module peut modifier la valeur de JL, et nous devrons donc procéder à des modifications : FOR...NEXT est remplacé par :

```
820 WK=1:REM BOUCLE PRINCIPALE
889 WK=WK+1:IF WK<=JL THEN 825
```



Les enregistrements

Nous abordons, cette fois en PASCAL, de nouvelles opérations sur des ensembles. Nous voyons également un autre type syntaxique, l'enregistrement.

Nous avons vu l'extrême utilité des ensembles ou jeux de caractères en PASCAL. Nous savons qu'ils s'implémentent très efficacement au niveau machine, du fait que leurs opérations ont des équivalents directs dans la plupart des jeux d'instructions inhérents aux processeurs. Un ensemble PASCAL s'obtient toujours en réunissant des éléments susceptibles d'être de tout type authentiquement scalaire (non réel). Toutes les opérations sur des ensembles portent sur la totalité de la structure, sauf pour l'opérateur IN, qui recherche une appartenance. Bien qu'il n'y ait pas d'opérateur de sélection d'un élément donné, nous pouvons utiliser les opérateurs d'inclusion dans un ensemble (<= et >=). Il suffit alors de tester l'équivalence des ensembles (leur « égalité »), par = et <>.

La limite inférieure d'un ensemble doit être une valeur ordinale de zéro ou plus. La limite supérieure a, impérativement, une certaine valeur absolue, entre 255 et 4095, selon l'implémentation. Remarquez que l'ensemble vide (représenté littéralement par []) est membre virtuel de tout ensemble, quel que soit son type. Cela peut sembler une faille dans la grande rigueur des types PASCAL, mais en pratique, le type d'un ensemble vide peut toujours être obtenu par déduction du type des autres ensembles d'une expression.

L'inclusion au sens strict de sous-ensembles authentiques ne peut être obtenue directement. Les opérateurs <and> ne sont pas définis pour les syntaxes d'ensembles. Cela est dû à des raisons d'implémentation. La plupart des restrictions du PASCAL sont justifiées du point de vue de la logique et de l'efficacité. Si vous voulez tester l'inclusion au sens strict, un test double est nécessaire, par exemple :

```
(A >= B) AND (A<>B)
```

Le test de la Maison, dans notre programme de Bingo, n'en avait pas besoin, un jeu de numéros appelés pouvant être soit équivalent (égal), soit, plus vraisemblablement, un sur-ensemble de celui constitué par les nombres de la carte. Ainsi, par exemple, cette expression booléenne aurait pu être écrite de la sorte :

```
Maison := Appelé > = Carte
```

qui devient *vrai* lorsque tous les éléments de l'ensemble Carte figurent dans celui des nombres appelés. Cette propriété fondamentale des ensembles contribue à faire de ces derniers des structures de données précieuses pour la résolution de problèmes. Actuellement, l'application la plus utile des ensembles est le test de sous-ensembles de type Char (caractère). Le modèle suivant (en français et en PASCAL) est approprié pour toute programmation interactive telle qu'un jeu.

```
Négative :=['N', 'n'];
Affirmative := ['O', 'o'];

REPEAT
   {Jouez}
   {affichage du (des) score(s)}
   write ('Encore ?');
   ReadLn (réponse);

   WHILE NOT (réponse IN Affirmative + Négative) DO
      BEGIN
         WriteLn ('O(ui) ou N(on)' :
         Colonnes);
         write ('Une autre partie ?');
         ReadLn (réponse)
      END

UNTIL réponse IN Négative
```

Lorsque nous voulons accéder à un élément d'une structure syntaxique, le choix demeure entre les tableaux (que nous connaissons bien), et les enregistrements. Ces derniers ont la propriété remarquable de refléter les enregistrements réels de données, qui comprennent des zones de tout type de données, simples ou structurées.

Utilisation pascal des enregistrements

La forme la plus courante d'enregistrement de données utilisée de manière professionnelle comporte des zones nom, adresse, numéro de téléphone, numéro de compte, etc. L'important est de pouvoir traiter l'information d'un enregistrement dans sa totalité, *mais également* de pouvoir accéder à une zone, en particulier pour effectuer les traitements voulus. Le PASCAL permet l'assignation de ces objets-enregistrements comme entités (ils peuvent aussi être manipulés autrement), ainsi que l'accès à leurs zones constitutives à des fins de comparaison ou de traitement selon leur type. La définition d'un enregistrement composite est très simple :

```
TYPE
   chambre = RECORD
      numéro : 1 ... 999;
      aile : (Nord, Est, Sud, Ouest);
      occupée : booléen
   END; {pièce}
VAR
   bureau : chambre;
```

De la même manière que l'instruction CASE était exceptionnelle en ce qu'elle employait le mot réservé END comme séparateur, la définition d'un enregistrement est la seule exception pour la partie déclaration d'un pavé de programme, exception à la règle qui veut que BEGIN et END soient toujours ensemble. C'est pourquoi il est utile d'indiquer la fin d'un enregistrement (END) par son identificateur, comme sur notre exemple. Toute variable de type chambre comporte trois zones. Dans le cas présent, elles sont de type scalaire différent, mais elles pourraient aussi bien être de même type, simple ou structuré. Il n'existe aucune restriction de quelque sorte aux types des zones d'enregistrements. Nous pourrions donc avoir une zone tableau de fichiers d'enregistrements contenant eux-mêmes des ensembles !

Entre les mots séparateurs RECORD et END, la syntaxe de définition est exactement la même que pour une déclaration VAR. Ici, cependant, nous déclarons des identificateurs de zones qui font partie de la structure de l'enregistrement. C'est pourquoi les noms numéro, aile et occupée ne peuvent figurer en dehors du champ défini par l'identificateur de l'enregistrement. Ces noms de zone sont des identificateurs locaux qui peuvent remplacer le nom des variables dans le programme. On y accède par les deux mécanismes PASCAL de sélection d'enregistrements : la notation dite « en pointillés », et l'instruction WITH.

Pour sélectionner une zone déterminée avec la première méthode, on utilise un point que l'on place entre l'identificateur de l'enregistrement et celui de la zone concernée. Par exemple :

```
bureau . numéro
```

reporte à la seule zone sous-ensemble de valeurs entières. Nous pourrions initialiser le contenu de l'enregistrement par des instructions telles que :

```
read (bureau . numéro)
'bureau . aile := Est;'
'bureau . occupée : = personne < >[ ]'
```

et ainsi de suite. Vous remarquez que nous avons glissé un ensemble vide d'un type dépendant de celui de personne — et que nous n'avons pas encore déclaré !

L'instruction WITH

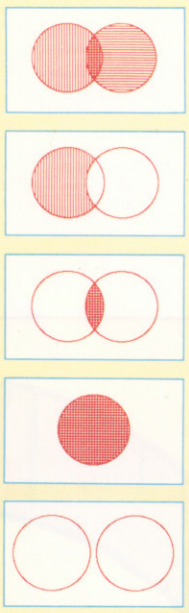
La notation dite « en pointillés » peut devenir lourde lorsque nous voulons accéder à la plupart ou à la totalité des zones d'un enregistrement. Une autre syntaxe indique seulement les identificateurs de zones. Sémantiquement parlant, la syntaxe de WITH signifie en gros : « J'ai besoin de cet enregistrement, et je ne donne que le nom des zones qui m'intéressent. » La syntaxe est exactement la même que celle de la boucle WHILE. La séquence des assignations d'initialisation que nous avons donnée pourrait avantageusement s'écrire :

```
WITH bureau DO
   BEGIN
      nombre : = 123;
      aile : = Est;
      occupée : = vrai
   END
```

Opérations pascal sur des ensembles

Voici la liste des opérations sur des ensembles disponibles avec le langage PASCAL (avec les diagrammes de Venn correspondants lorsqu'il y a lieu).

1. Union (S1 + S2) : donne le sur-ensemble comprenant tous les objets des ensembles S1 et S2 réunis (diagramme 1).
2. Différence (S1 – S2) : sous-ensemble constitué des éléments de S1 qui n'appartiennent pas à S2 (diagramme 2).
3. Intersection (S1 * S2) : sous-ensemble réunissant les éléments communs aux deux ensembles (diagramme 3).
4. Équivalence (S1 = S2) : identité des deux ensembles S1 et S2 dotés des mêmes objets (diagramme 4).
5. Non-équivalence (S1 < > S2) : vrai lorsque tous les éléments de S1 sont distincts de ceux de S2, et vice versa (diagramme 5).
6. Inclusion (S1 < = S2) : lorsque tout élément de S1 appartient aussi à S2.
7. Inclusion (S1 > = S2) : lorsque tout élément de S2 appartient aussi à S1.
8. Appartenance (m IN S1) : vrai lorsque l'ensemble à un seul élément ([m]) est un sous-ensemble de S1.



Liz Dixon

Le meilleur coup

Voici notre solution au problème posé la fois précédente, où nous vous demandions d'écrire un programme calculant le nombre maximal de coups à la suite au cours d'une partie de billard. (Cl. Ian McKinnell.)

```
PROGRAMME   CoupMax  (sorties);
CONST
    BoulesRouges = 15;
TYPE
   billard = (blanc, rouge, jaune, vert,
               marron,  bleu,  rose,  noir );
VAR
       boule : billard;
       score : BoulesRouges .. MaxInt;
BEGIN
   score  :=  BoulesRouges * (  ord (rouge ) + ord (noir) );
   FOR boule := jaune TO noir DO
       score := score + ord ( boule );
   WriteLn  (  "le  nombre  maximum  de  coups
                  est : ", score : 3)
END
```

Ian McKinnell

Dans tout le champ de l'instruction WITH (ici, depuis BEGIN jusqu'à END), les identificateurs de zone n'ont pas à mentionner l'identificateur d'enregistrement correspondant suivi du pointillé, et peuvent être indiqués directement comme plus haut. Aucune confusion n'est possible, même si une autre variable du programme avait par exemple le nom Numéro. Le champ local a toujours priorité. La notation avec pointillés per-

met de faire passer des valeurs dans le champ de l'identificateur local. Par exemple :

```
bureau . numéro : = Numéro
```

assignerait une valeur d'une variable externe (Numéro), à la zone de l'enregistrement (numéro). Souvenez-vous que le PASCAL ne distingue pas les majuscules des minuscules, et que l'identificateur non qualifié nombre renvoie à une variable externe lorsqu'il n'est pas dans une instruction WITH. Il serait de toute manière essentiel d'utiliser la notation pointillée pour une affectation entre deux variables de même type :

```
bagages . aile : = réception . aile
```

L'instruction WITH ne peut être utilisée ici que pour indiquer une zone de la variable. Dans le cas contraire, il y aurait ambiguïté, et le compilateur ne l'accepterait pas. La meilleure syntaxe serait :

```
WITH bagages DO
    aile : = réception . aile
```

Il en résulte que les deux notations ont leur utilité. C'est l'application qui permet de dire quelle construction utiliser.

Longueurs

Une définition de type enregistrement est utilisée pour attribuer une zone séparée pour chaque unité de longueur et pour chaque valeur. Le programme ajoute les longueurs, reportant les retenues de chaque unité sur les unités supérieures. Il utilise pour cela les opérateurs d'entiers DIV et MOD pour obtenir respectivement la somme et le reste. Les résultats sont assignés aux zones de « Total » et affichés. L'affectation directe d'enregistrements entiers serait plus rapide (c'est-à-dire Alongueur : = Blongueur), mais chaque traitement doit concerner chaque zone en particulier, de sorte que Total : = Alongueur + Blongueur est illégal. Remarquez l'utilisation de la notation pointillée, et celle de l'instruction WITH pour lire les valeurs de Alongueur et Blongueur.

```
PROGRAMME Longueurs ( entrées, sorties);
CONST
    OctetMax  = 255;
TYPE
    octet = 0 .. OctetMax;
    longueur = RECORD
                   cm : 0 .. 9;
                   dm : 0 .. 9;
                   m : octet
               END; {longueur}

VAR
    cm,
   dm  : octet;
    Alongueur,
    Blongueur,
    Total  : longueur;
BEGIN
  WriteLn  ( 'frappez  une  longueur  en  m, dm, et cm, ' );
  WriteLn  ( 'séparés  par  une espace ou par
                  RC.' );
  WriteLn;
  write ( 'Premier : ' :        );
  read  (  Alongueur  .  m,  Alongueur  . dm,
                Alongueur . cm );
  write ( 'Deuxième : ' :      );
  WITH Blongueur DO
  read ( m,dm, cm );
cm := Alongueur . cm + Blongueur . cm;
dm :=  Alongueur . dm + Blongueur  . dm  +  cm
       DIV 10;
  WriteLn;
  WITH total DO
    BEGIN
      cm := cm MOD 10;
     dm := dm MOD 10;
      m  := Alongueur  .  m  +  Blongueur  .
                m  +dm DIV 10;
  WriteLn  ( 'La  longueur  totale  est
                :' );
  WriteLn  (  m  :  , ' m,  ', dm : ,
              ' dm,  et ', cm : ,  ' cm.')

  END
END
```

Protection des données

Nous vous avions posé, à propos du programme sur le Bingo, le problème de l'écriture de boucles dans le programme destinées à empêcher la répétition d'un nombre déjà appelé, et à détecter tout nombre en dehors de la fourchette 0 à 90. La meilleure façon de protéger le programme de Bingo de saisies de données illégales est d'ajouter une syntaxe WHILE après les deux instructions ReadLn. La structure est la même dans chaque cas :

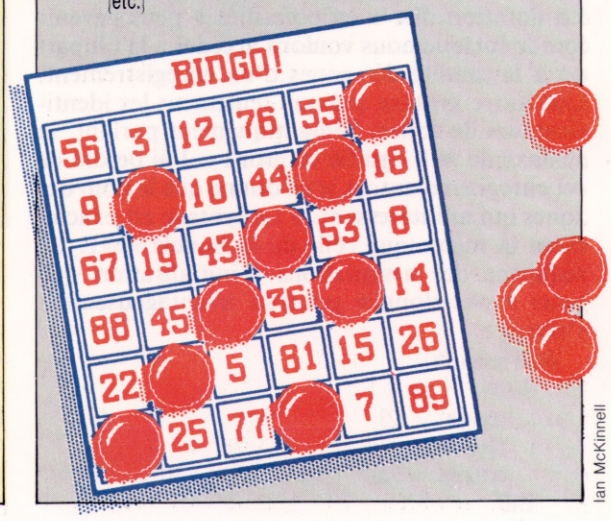
```
WHILE le nombre est inacceptable DO
    afficher un message d'erreur approprié
    afficher un nouveau message-guide
    opérateur
    lire les données à nouveau
```

Pour la saisie initiale des chiffres devant remplir les cartes, nous devons nous assurer que les nombres tapés ne sont pas hors champ (de 0 à 90). Nous devons également empêcher la répétition d'un nombre déjà tapé. Ainsi :

```
WHILE NOT (nombre IN Permis) OR (nombre IN Carte) DO
    Begin
        WriteLn (nombre : 20, 'illégal');
        write ('Tapez à nouveau' : 14);
        ReadLn (nombre)
    END;
      {etc.}
```

La même stratégie s'applique à l'appel des nombres, la boucle de validation devant néanmoins s'écrire :

```
WHILE NOT (nombre IN Permis) OR (nombre IN Appelé) DO
    {etc.}
```



Ian McKinnell

Autour du bloc

Pour commencer notre investigation des aspects utiles ou inhabituels du système d'exploitation du Commodore 64, nous examinons le premier bloc mémoire de 1 K.

La première chose qu'il faut savoir sur un ordinateur, c'est : « Quelle est la structure générale de la carte d'implantation mémoire? » Dans le cas du Commodore 64, la réponse est : le microprocesseur 6510 peut « voir » une table parmi *plusieurs,* suivant la configuration que le programmeur confère à la machine.

C'est parce que le 6510 est un microprocesseur très souple, capable de commuter différents blocs de mémoire. Ce peut être fait sur le matériel, en mettant les broches 8 et 9 sur le port d'extension haut (+5 V) ou bas (0 V), ou sur le logiciel, en modifiant le contenu de l'adresse 1. Ces manœuvres modifieront toutes deux radicalement la manière dont le 6510 voit la mémoire.

Nous schématisons la table normale (par défaut), qui montre les 64 K de mémoire normalement vus par le processeur 6510. Une analyse plus détaillée de la table d'implantation est donnée dans le tableau.

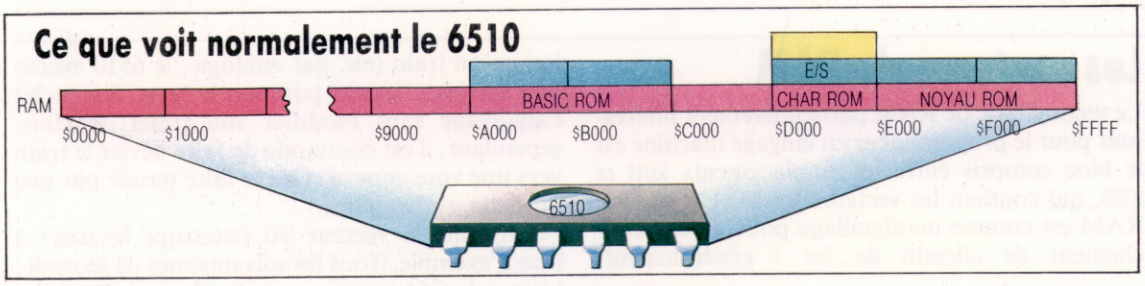
Pour le moment, supposons que le C64 soit en mode par défaut. La première chose à noter est que l'ordinateur a 64 K de RAM et 20 K de ROM. A aucun moment, le 6510 ne peut adresser plus de 64 K sur le total de 84 K, d'où la nécessité de commutation de bancs mémoire. Le diagramme d'implantation mémoire montre que l'interpréteur BASIC ROM — le programme en langage machine qui exécute les programmes BASIC — est placé entre $A000 et $CFFF, et le noyau ROM — le code qui manie toutes les fonctions d'entrée/sortie — entre $E000 et $FFFF.

Au-dessous de ces deux blocs de ROM, il y a deux blocs de 8 K de RAM, que le 6510 ne voit normalement pas, quoiqu'un POKE (ou STA) permette d'accéder à la RAM sous-jacente. Un PEEK donnera le contenu de la ROM, mais la RAM peut être lue en sortant le bloc de ROM correspondant.

Il faut se rappeler que, si l'on veut utiliser les commandes BASIC PEEK ou POKE, il vaut mieux ne pas sortir le bloc de ROM BASIC. De même que tout programme BASIC nécessite une RAM pour y stocker ses variables, de même l'interrupteur et les routines du noyau ont besoin du SE RAM (Système d'exploitation mémoire vive).

Le SE RAM est situé entre $0002 et $03FF, et une de ses composantes importantes, la pile, occupe les emplacements de $0100 à $01FF. Le SE RAM est placé à la page zéro parce que c'est la partie de table d'implantation qui permet l'accès le plus rapide à la RAM, et la vitesse d'exécution est une considération importante pour le système d'exploitation (SE).

Si l'on veut interfacer un programme BASIC avec un code machine, il est alors essentiel de savoir comment fonctionne cette zone de la RAM. Considérons deux aspects importants du SE RAM : les pointeurs BASIC et les vecteurs code machine.



ROM avec vue
Comme le processeur 6510 utilise des adresses 16 bits, il ne peut adresser qu'un maximum de 65 536 (64 K) emplacements mémoire. Le C64 a un total de 84 K de ROM et RAM, mais ne peut « voir » que 64 K à la fois, en commutant des blocs de mémoire de sa table d'implantation en mémoire.
Le premier « bloc » représente la mémoire que voit normalement le 6510. Les zones placées derrière lui sont accessibles en plaçant des registres spéciaux de blocs. Par exemple, si un programme est écrit purement en langage machine, alors la ROM BASIC peut ne pas être nécessaire et est exclue, de sorte qu'on peut utiliser la RAM située derrière elle.

Table d'implantation en mémoire du C64

Emplacements	Utilisation
$0000-$0001	Registres de contrôle de table d'implantation 6510.
$0002-$03FF	RAM Système d'exploitation.
$0400-$07F7	RAM Écran.
$07F8-$07FF	Pointeurs de lutins.
$0800-$9FFF	Zone programme BASIC utilisateur (variables incluses).
$A000-$BFFF	ROM interpréteur BASIC.
$C000-$CFFF	RAM libre de 4 K.
$D000-$D02E	Registres de contrôle de puce VIC II (6566/9).
$D02F-$D3FF	Images VIC II.
$D400-$D41C	Registres de contrôle de puce SID (6581).
$D41D-$D7FF	Images SID.
$D800-$DBE7	Demi-octets couleur (4 bits chacun).
$DBE8-$DBFF	Demi-octets inutilisés.
$DC00-$DC0F	Registres de contrôle E/S CIA # 1 (6526).
$DC10-$DCFF	Images CIA # 1.
$DD00-$DD0F	Registres de contrôle E/S CIA # 2 (6526).
$DD10-$DDFF	Images CIA # 2.
$E000 -$FFFF	ROM noyau.

Pointeurs basic

Modifier le contenu des pointeurs BASIC peut être parfois bien utile. Par exemple, les contenus normaux de $002B (43 décimal) et $002C (44 décimal) sont respectivement 1 et 8. C'est l'adresse de début de BASIC sous la forme octet lo/octet hi, c'est-à-dire que BASIC commence en 8 × 256 + 1 = 2049, soit $0801. En fait, il commence en $0800, mais le SE requiert un premier octet toujours nul, de sorte que le programme BASIC proprement dit commence en $0801. (Signalons que $0801 est l'adresse à laquelle on commence à sauvegarder un programme BASIC à l'aide du moniteur code machine Commodore.)

Avant de charger un programme BASIC, nous pouvons modifier le bas de BASIC en manipulant ces deux pointeurs, en prenant garde à s'assurer que BASIC commence à une limite de page (c'est-à-dire conserve le contenu de l'emplacement $002B à 1) et commence par un zéro. (Notez aussi qu'une « page » est un bloc de 256 octets de mémoire.) Ainsi :

```
POKE 2560,0 : POKE 44,10 : NEW
```

exécuté en mode direct fera monter le bas de la mémoire de deux pages jusqu'à $2560. Ici, on utilise NEW pour réinitialiser rapidement tous les pointeurs (qui occupent les emplacements $002D à $0038). Faire monter le bas de BASIC peut servir à contenir simultanément deux programmes BASIC en mémoire. Il suffit de charger le premier programme, faire monter le bas de BASIC, puis charger le second.

Mais, plus souvent, on aura besoin d'abaisser le haut de la mémoire pour faire de la place pour un programme en langage machine. Ainsi :

```
POKE 56,159 : POKE 51,0 : POKE 52,159
```

abaissera le haut de la mémoire BASIC d'une page.

Une fois que nous avons déplacé un bloc de RAM de la zone BASIC, nous pouvons être sûrs que le SE n'utilisera pas cette zone pour stocker des variables BASIC, et c'est pourquoi notre langage machine sera sauf... erreurs !

Table de pointeurs basic CBM 64

Emplacements	Pointeurs
$002B-$002C	Début de BASIC utilisateur
$002D-$002E	Début de variables BASIC
$002F-$0030	Début de tableaux BASIC
$0031-$0032	Fin de tableaux BASIC + 1
$0033-$0034	Bas des chaînes
$0035-$0036	Utilitaires
$0037-$0038	Fin de BASIC utilisateur

Les vecteurs de RAM

Le second bloc de RAM particulièrement intéressant pour le programmeur en langage machine est le bloc compris entre les emplacements $0314 et $0333, qui contient les vecteurs RAM. Un vecteur RAM est comme un aiguillage pour un embranchement de chemin de fer : généralement, lorsqu'un train (ou, par analogie, le 6510 exécutant son programme) parcourt la ligne, il franchit l'aiguillage sans modifier son trajet. Parfois, cependant, il est commode de faire dévier le train vers une voie annexe et de le faire passer par une ou deux autres gares.

Utilisons le vecteur IRQ (Interrupt ReQuest) à titre d'exemple. Tous les soixantièmes de seconde, lorsque le C64 fonctionne normalement, l'une des horloges de la puce E/S 6526 est basculée et abaisse la ligne IRQ 6510. A la fin de son instruction en cours, le 6510 répond à l'abaissement de la ligne IRQ en générant une interruption et en commençant la routine de service IRQ, partie de code qui débute en $FF48. Entre autres choses, la routine de service parcourt le clavier pour détecter si une touche a été appuyée. L'une des premières choses que fait cette routine est :

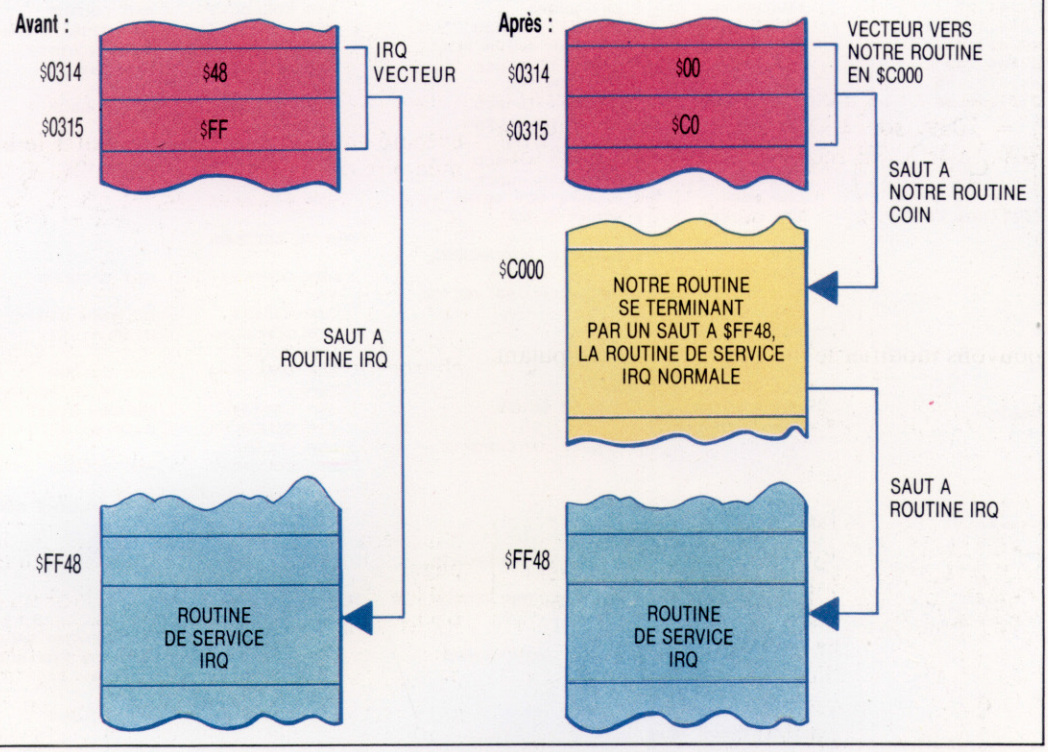
```
JMP ($0314)
```

c'est-à-dire faire un saut indirect à l'adresse donnée en $0314 (octet lo) et $0315 (octet hi). Ces deux octets sont en RAM, et il paraît raisonnable de penser que, si nous changeons leur contenu, nous pouvons pointer vers leur propre partie de code. Après avoir exécuté notre code, nous pouvons repointer le microprocesseur dans la direction normale. Ainsi, tant que notre code n'est pas trop long (ce qui exclut l'usage de toute routine de noyau), nous pouvons faire effectuer au 6510 une petite routine tous les soixantièmes de seconde. On appelle « coin » une telle partie de code.

Table de vecteurs CBM 64

Emplacements	Vecteur
$0314-$0315	Interruption IRQ
$0316-$0317	Interruption BRK
$0318-$0319	Interruption NMI
$031A-$031B	Routine noyau OPEN
$031C-$031D	Routine noyau CLOSE
$031E-$031F	Routine noyau CHKIN
$0320-$0321	Routine noyau CHKOUT
$0322-$0323	Routine noyau CLRCHIN
$0324-$0325	Routine noyau CHRIN
$0326-$0327	Routine noyau CHROUT
$0328-$0329	Routine noyau STOP
$032A-$032B	Routine noyau GETIN
$032C-$032D	Routine noyau CLALL
$032E-$032F	Défini par l'utilisateur
$0330-$0331	Routine noyau LOAD
$0332-$0333	Routine noyau SAVE

En introduisant un coin



Le coin du temps
Le processeur 6510 est interrompu tous les soixantièmes de seonde pour effectuer les tâches de service, telles que l'examen du clavier. L'adresse de départ de la routine est contenue dans deux emplacements : $0314 et $0315.
En changeant l'adresse contenue dans ces emplacements, nous pouvons « coincer » notre propre partie de code, qui sera exécutée à la place de cette routine. De la sorte, nous pouvons nous assurer que notre code sera exécuté tous les soixantièmes de seconde, lorsqu'une interruption est générée.
(Cl. Kevin Jones.)

Horloge Commodore

Le listage d'assemblage suivant et le programme chargeur BASIC démontrent les principes d'insertion d'une partie de code « coin » dans la routine d'interruption normale IRQ. Le coin fait marcher une horloge dans un angle de l'écran. Lorsque l'horloge est mise à l'heure par l'interruption normale, qui a lieu tous les soixantièmes de seconde, la machine fonctionne normalement, l'horloge marchant selon un programme BASIC à taper et exécuter simultanément. La routine établit d'abord le temps passé, qui modifie les contenus de $0314 et $0315 pour indiquer le code coin, qui fait avancer et affiche l'horloge chaque fois qu'une interruption IRQ est générée. Enfin, on fait un saut (JMP) vers la routine de service IRQ normale, dont l'adresse est conservée dans les emplacements VECTOR et VECTOR+1.
Au lieu de taper et d'assembler ce code source, il est aussi possible d'entrer le programme sous la forme d'une série d'instructions DATA, qui constituent le programme en langage machine. Tapez simplement le programme chargeur BASIC et exécutez-le (RUN), pour voir apparaître l'horloge dans le coin supérieur droit de l'écran. Un total de contrôle est inclus, pour s'assurer que tous les DATA ont été correctement entrés. Si le programme s'arrête avec le message « CHECKSUM ERROR », vérifiez si vous n'avez pas fait d'erreur dans vos instructions DATA.
L'horloge peut être réglée en POKEant l'instant de départ en heures, minutes et secondes dans les emplacements 50129, 50130 et 50131. A la fin du programme chargeur BASIC, l'horloge est réglée par les lignes 1480 et suivantes, pour commencer à 4:30:00 de l'après-midi. Il suffit de changer les valeurs POKEées dans ces emplacements pour modifier l'heure de départ en conséquence. SYS 50138 provoque l'exécution du programme en langage machine et le démarrage de l'horloge. Celle-ci peut être arrêtée en appuyant simultanément sur les touches « Run/Stop » et « Restore », ou en exécutant une routine spéciale d'arrêt incluse dans le programme en langage machine en entrant SYS 50237.

Chargeur basic

```
1000 REM ** CLOCK BASIC LOADER **
1010 DATA173,166,2,240,10,169,128,13,14
1020 DATA221,141,14,221,48,8,169,127,45
1030 DATA14,221,141,14,221,169,127,45
1040 DATA15,221,141,15,221,173,208,195
1050 DATA41,128,141,208,195,173,209,195
1060 DATA32,28,197,13,208,195,141,11
1070 DATA221,173,210,195,32,28,197,141
1080 DATA10,221,173,211,195,32,28,197
1090 DATA141,9,221,169,0,141,8,221,120
1100 DATA173,20,3,141,214,195,173,21,3
1110 DATA141,215,195,169,76,141,20,3
1120 DATA169,196,141,21,3,88,96,120,173
1130 DATA214,195,141,20,3,173,215,195
1140 DATA141,21,3,88,96,173,216,195,201
1150 DATA6,240,3,76,8,197,169,255,141
1160 DATA216,195,173,213,195,240,243
1170 DATA173,11,221,170,41,128,208,5
1180 DATA169,1,76,111,196,169,16,141,38
1190 DATA4,173,212,195,141,38,216,163
1200 DATA13,141,39,4,173,212,195,141,39
1210 DATA216,138,41,16,32,14,197,141,28
1220 DATA4,173,212,195,141,28,216,138
1230 DATA32,22,197,141,29,4,173,212,195
1240 DATA141,29,216,169,58,141,30,4,173
1250 DATA212,195,141,30,216,173,10,221
1260 DATA170,32,14,197,141,31,4,173,212
1270 DATA195,141,31,216,138,32,22,197
1280 DATA141,32,4,173,212,195,141,32
1290 DATA216,169,47,141,33,4,173,212
1300 DATA195,141,33,216,173,9,221,170
1310 DATA32,14,197,141,34,4,173,212,195
```

```
1320 DATA141,34,216,138,32,22,197,141
1330 DATA35,4,173,212,195,141,35,216
1340 DATA169,46,141,36,4,173,212,195
1350 DATA141,36,216,173,8,221,105,48
1360 DATA141,37,4,173,212,195,141,37
1370 DATA216,238,216,195,108,214,195,74
1380 DATA74,74,74,24,105,48,96,41,15,24
1390 DATA105,48,96,160,255,56,200,233
1400 DATA10,176,251,105,10,141,217,195
1410 DATA152,10,10,10,10,13,217,195,96
1420 DATA42131:REM*CHECKSUM*
1430 CC=0
1440 FORI=50138TO50481
1450 READX:CC=CC+X:POKEI,X
1460 NEXT
1470 READX:IFCC<>XTHENPRINT"CHECKSUM ERROR"
1480 REM** TEST CLOCK **
1490 POKE50128,128:REM AM/PM
1500 POKE50132,8:REM COLOUR
1510 POKE50133,1:REM DISPLAY
1520 POKE50129,4:REM HOURS
1530 POKE50130,30:REM MINUTES
1540 POKE50131,0:REM SECONDS
1550 SYS50138:REM CALL ROUTINE

TO TURN OFF THE WEDGE USE SYS50237
IN DIRECT OR PROGRAM MODE.
```

Listage d'assemblage

```
;**********************************
;*                                *
;*        CLOCK IRQ WEDGE         *
;*                                *
;* 50128 = AM=0 / PM=128          *
;* 50129 = HOURS                  *
;* 50130 = MINUTES                *
;* 50131 = SECONDS                *
;* 50132 = CLOCK COLOUR           *
;* 50133 = DISPLAY ON=1/OFF=0     *
;*                                *
;* WEDGE INSERT  SYS 50138        *
;* WEDGE REMOVE  SYS 50237        *
;*                                *
;**********************************

IRQVEC = $0314   ; IRQ RAM VECTOR
CLOCK = $DD08    ; TOD REGISTER
D2CRA = $DD0E    ; VIA#2 CRA
D2CRB = $DD0F    ; VIA#2 CRB
PALNTS = $02A6   ; PAL/NTSC FLAG
RATE = $06       ; DISPLAY EVERY 6 IRQS
DIGIT = $30      ; SCREEN CODE FOR '0'
POINT = $2E      ; SCREEN CODE FOR '.'
SLASH = $2F      ; SCREEN CODE FOR '/'
COLON = $3A      ; SCREEN CODE FOR ':'
HZ50 = $80       ; 50 HZ MASK FOR TODIN
HZ60 = $7F       ; 60 HZ MASK FOR TODIN
AY = $01         ; SCREEN CODE FOR 'A'
PEE = $10        ; SCREEN CODE FOR 'P'
EM = $0D         ; SCREEN CODE FOR 'M'
WRITE = 127      ; MASK TO SET CLOCK IN CRB
SCNLOC = $041C   ; CLOCK ADDR ON SCREEN
COLLOC = $D81C   ; ADDR ON VIDEO MATRIX
TRNCLO = $0F     ; MASK FOR LOW NYBBLE

* = $C3D0         ; START ADDRESS CODE
AMPM *=*+1       ; AM/PM FLAG
HOURS *=*+1      ; HOURS VAL (FOR INITIALISE)
MINS *=*+1       ; MINUTES VALUE
SECS *=*+1       ; SECONDS VALUE
COLOR *=*+1      ; CLOCK COLOUR
DISPLY *=*+1     ; DISPLAY/HIDE FLAG
VECTOR *=*+2     ; STORAGE FOR OLD IRQ VEC
COUNT *=*+1      ; IRQ COUNTER
TEMP1 *=*+1

;
;INSERT WEDGE
;
   LDA PALNTS       ; PAL OR NTSC
   BEQ NTSC         ; BRANCH FOR NTSC
   LDA #HZ50        ; MUST BE PAL
   ORA D2CRA
   STA D2CRA        ; SET TOSIN FOR 50 HZ
   BMI PALDUN
NTSC
   LDA #HZ60        ; NTSC
   AND D2CRA
   STA D2CRA        ; SET TODIN FOR 60 HZ
PALDUN
   LDA #WRITE
   AND D2CRB        ; SET CLOCK NOT ALARM
   STA D2CRB
   LDA AMPM
   AND #128         ; MAKE AMPM VALUE VALID
   STA AMPM
   LDA HOURS        ; GET HOURS
   JSR BINBCD       ; CONVERT TO BCD
   ORA AMPM         ; OR WITH AM/PM FLAG
   STA CLOCK+3      ; STORE IN CLOCK
   LDA MINS         ; GET MINUTES
   JSR BINBCD       ; CONVERT TO BCD
   STA CLOCK+2      ; STORE IN CLOCK
   LDA SECS         ; GET SECONDS
   JSR BINBCD       ; CONVERT TO BCD
   STA CLOCK+1      ; STORE IN CLOCK
   LDA #00          ; ALWAYS SET 10THS TO 0
   STA CLOCK        ; START CLOCK
;
   SEI              ; DISABLE INTERRUPTS
   LDA IRQVEC
   STA VECTOR       ; SAVE OLD IRQ VECTOR
   LDA IRQVEC+1
   STA VECTOR+1
;
   LDA #<WEDGE
   STA IRQVEC
   LDA #>WEDGE      ; INSERT WEDGE
   STA IRQVEC+1
   CLI              ; ENABLE INTERRUPTS
   RTS

;
;REMOVE WEDGE
;
   SEI              ; DISABLE INTERRUPTS
   LDA VECTOR
   STA IRQVEC       ; RESTORE RAM VECTOR
   LDA VECTOR+1
   STA IRQVEC+1
   CLI              ; ENABLE INTERRUPTS
   RTS

;
;WEDGE STARTS HERE
;
WEDGE
   LDA COUNT        ;
   CMP #RATE        ; DO CLOCK THIS IRQ?
   BEQ CONT         ;
OUT
   JMP EXIT         ; NO
CONT
   LDA #$FF         ; RESET IRQ COUNTER
   STA COUNT
   LDA DISPLY       ; DISPLAY?
   BEQ OUT          ; NO...BRANCH

   LDA CLOCK+3      ; GET HOURS/AM/PM
   TAX              ; PUT A COPY IN X REG
   AND #$80         ; GET AM/PM
   BNE PM           ; BRANCH IF PM
   LDA #AY          ; DISPLAY 'A'
   JMP MERIDP
PM
   LDA #PEE         ; DISPLAY 'P'
MERIDP
   STA SCNLOC+10
   LDA COLOR        ; GET COLOUR
   STA COLLOC+10    ; SET COLOUR
   LDA #EM          ; DISPLAY 'M'
   STA SCNLOC+11
   LDA COLOR        ; SET COLOUR
   STA COLLOC+11
;
;DO HOURS
;
   TXA              ; GET HOURS
   AND #$10         ; JUST WANT HIGH DIGIT
   JSR HIDIGT       ; GET SCREEN CODE
   STA SCNLOC       ; DISPLAY IT
   LDA COLOR
   STA COLLOC       ; SET COLOUR
   TXA              ; GET BYTE AGAIN
   JSR LODIGT       ; GET LOW DIGIT
   STA SCNLOC+1     ; DISPLAY IT
   LDA COLOR
   STA COLLOC+1     ; SET COLOUR
;
   LDA #COLON       ; HRS/MINS SEPARATOR
   STA SCNLOC+2
   LDA COLOR
   STA COLLOC+2
;
;NOW DO MINUTES
;
   LDA CLOCK+2      ; GET MINUTES
   TAX
   JSR HIDIGT       ; DO HIGH DIGIT
   STA SCNLOC+3     ; DISPLAY IT
   LDA COLOR
   STA COLLOC+3     ; AND COLOUR
   TXA              ; GET BYTE AGAIN
   JSR LODIGT       ; DO LOW DIGIT
   STA SCNLOC+4     ; DISPLAY IT
   LDA COLOR
   STA COLLOC+4     ; SET COLOUR
;
   LDA #SLASH       ; MIN/SEC SEPARATOR
   STA SCNLOC+5
   LDA COLOR
   STA COLLOC+5
;
;NOW DO SECONDS
;
   LDA CLOCK+1      ; GET SECONDS
   TAX
   JSR HIDIGT       ; DO HIGH DIGIT
   STA SCNLOC+6     ; DISPLAY IT
   LDA COLOR
   STA COLLOC+6     ; AND COLOUR
   TXA              ; GET BYTE AGAIN
   JSR LODIGT       ; DO LOW DIGIT
   STA SCNLOC+7     ; DISPLAY IT
   LDA COLOR
   STA COLLOC+7     ; AND COLOUR
;
   LDA #POINT       ; SECS/TENTHS SEPARATOR
   STA SCNLOC+8
   LDA COLOR
   STA COLLOC+8
;
;NOW DO TENTHS
;
   LDA CLOCK        ; GET TENTHS VALUE
   ADC #DIGIT       ; ADD $30 FOR SCREEN CODE
   STA SCNLOC+9     ; DISPLAY IT
   LDA COLOR
   STA COLLOC+9     ; AND COLOUR
;
;
EXIT
   INC COUNT        ; INCREMENT IRQ COUNTER
   JMP (VECTOR)     ; GO TO REST OF IRQ

;
;SUBROUTINES
;
HIDIGT
   LSR A
   LSR A            ; MOVE HIGH NYBBLE INTO LOW
   LSR A
   LSR A
   CLC
   ADC #DIGIT       ; ADD $30 FOR SCREEN CODE
   RTS

LODIGT
   AND #TRNCLO      ; MASK OFF HIGH NYBBLE
   CLC
   ADC #DIGIT       ; ADD $30 FOR SCREEN CODE
   RTS

;
;CONVERT BINARY TO BCD
;
BINBCD
   LDY #$FF
   SEC
D10
   INY
   SBC #10          ; SUBTRACT 10 UNTIL -VE
   BCS D10
   ADC #10          ; ADD 10 BACK ON
   STA TEMP1        ; STORE REMAINDER
   TYA              ; GET NUMBER PF 10S SUBTRACTED
   ASL A
   ASL A
   ASL A            ; SHIFT INTO HIGH NYBBLE
   ASL A
   ORA TEMP1        ; PUT REMAINDER IN LOW NYBBLE
   RTS
```

N.B. : les lignes REM du programme Chargeur BASIC peuvent être respectivement traduites par CHARGEUR BASIC D'HORLOGE et ROUTINE D'APPEL.

Programme reproduit avec l'aimable autorisation des auteurs et Ellis Horwood Ltd. D'après *Mastering the Commodore 64* par Jones et Carpenter.

Page manquante
(publicité)

Page manquante
(publicité)